BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 65, No. 3, 2017
DOI: 10.1515/bpasts-2017-0036

Decomposition of multi-output functions oriented to configurability of logic blocks

M. KUBICA[1]* and D. KANIA[2]

[1]Faculty of Mechanical Engineering and Computer Science, University of Bielsko-Biała, 2 Willowa St., 43-309 Bielsko-Biała, Poland
[2]Institute of Electronics, Silesian University of Technology, 2A Akademicka St., 44-100 Gliwice, Poland

Abstract. The main goal of the paper is to present a logic synthesis strategy dedicated to an LUT-based FPGA. New elements of the proposed synthesis strategy include: an original method of function decomposition, non-disjoint decomposition, and technology mapping dedicated to configurability of logic blocks. The aim of all of the proposed synthesis approaches is the sharing of appropriately configured logic blocks. Innovation of the methods is based on the way of searching decomposition, which relies on multiple cutting of an MTBDD diagram describing a multi-output function. The essence of the proposed algorithms rests on the method of unicoding dedicated to sharing resources, searching non-disjoint decomposition on the basis of the partition of root tables, and choosing the levels of diagram cutting that will guarantee the best mapping to complex logic blocks. The methods mentioned above were implemented in the MultiDec tool. The efficiency of the analyzed methods was experimentally confirmed by comparing the synthesis results with both academic and commercial tools.

Key words: BDD, decomposition, logic synthesis, technology mapping.

1. Introduction

Configurable logic resources of logic blocks, included in LUT-based FPGA structures, enable carrying out any function that has a specified number of variables. The number of inputs of logic blocks is considerably low (4–6), and that is why one of the main problems of logic synthesis is decomposition. It is the theoretical background of the partition of function variables. The result of function decomposition is the partition of a designed circuit on LUT blocks that have a given number of inputs.

The most popular model of function decomposition is the model suggested by Ashenhurst and Curtis [1, 2]. In fact, the main aim of this model was to transform Boolean expressions. Besides, it was competitive for the minimization method created by Quine McCluskey. The creation of LUT-based FPGA structures arouses interest in the theory of decomposition. A series of various types of algorithms have been created from scratch and were adopted from other synthesis strategies dedicated to gate arrays (MIS-PGA [3], ASYL [4], Chortle [5], etc.)

Function decomposition is strongly associated with technology mapping in the blocks of an LUT-based type that have a given number of inputs. Technology mapping, which was used in many approaches, is connected with the problem of multi-level minimization. The essence of technology mapping is based on appropriate logic net partition. In a very early stage, it often corresponds to the function after a two-level minimization. Multi-level optimization focuses on the factorization of Boolean functions and leads to the separation of common elements carried out by shared logic resources. It often happens that factorization procedures are supported by the procedures of variables ordering, searching for core functions, and various strategies of connecting separated elements. In fact, the essence of technology mapping is the partition and converting of a logic net that describes Boolean function in general, together with an appropriate choice of nodes or branches reflecting a multi-level form of function [5–12]. Configurability of logic blocks, for instance ALMap [13], is more and more often used in the process of technology mapping. In the process of multi-level optimization, various types of graphs, such as BDD, are used [14–17].

There are the methods of function technology mapping in LUT-based FPGA structures in which a classic model of decomposition plays a crucial role. A classic model of decomposition may also be one of the elements of synthesis dedicated to CPLD structures [18–20]. In fact, it is the basis of function synthesis reflected in LUT-based FPGA structures. Sometimes, logic synthesis method is targeted at FPGA architectures with specialized embedded memory blocks (EMB) [21, 22]. In all cases, decomposition of Boolean functions turns out to be a key problem. Decomposition of multi-output functions leads to better results than decomposition done separately on each single function. It is because of the fact that shared blocks are the result of decomposition of Boolean functions and are used by several single-output functions at the same time. A classic model of decomposition is the basis of a series of decomposition algorithms [5, 23–25]. Synthesis methods use various graph algorithms that enable indication of column multiplicity of partition tables or a proper transformation of Boolean functions. Undoubtedly, one of the main advantages of a classic function decomposition is the natural admitting of don't-care states. It is probably the most important element making these methods one of the most effective. In last years, a series of various synthesis

*e-mail: mkubica@ath.bielsko.pl

systems, which also give satisfying results, have been created. The examples of such systems include BDS [26], BDS-PGA 2.0 [17], DDBDD [27, 28], and DAOmap [7]. The elements of resynthesis are more and more often used in the process of synthesis [8, 29, 30]. ABC system turns out to be especially vital as it uses AIG graphs in the process of complex synthesis of combinational, as well as sequential circuits [31].

The methods using binary decision diagrams are of particular interest [15, 26–28, 32–36]. The most important advantage is the relatively low calculating, as well as memory complexity of the algorithms using BDD. Taking everything into account, it is crucial to search for effective function decomposition that would be described using binary decision diagrams. Besides, it is especially beneficial to direct the searching of the partition of BDD diagram matched to the configurability of logic blocks.

The purpose of the paper is to present new elements of function decomposition whose main aim is to guarantee effective technology mapping in LUT-based logic blocks. Function decomposition is the basis of technology mapping matched to configurability of logic blocks. Original technology mapping approach is proposed too.

2. Theoretical background

Let f be n-input in m-output logic function reflecting B^n set into B^m set i.e. $f: B^n \to B^m$, where $B = \{0, 1\}$. Function $f: B^n \to B^m$ may be presented as $Y = f(I_{n-1}, ..., I_1, I_0)$, where $Y = \{y_{m-1}, ..., y_1, y_0\}$. Function $f: B^n \to B^m$ is subjected to decomposition, i.e.

$$f(X_f, X_b) = F[g_1(X_b), g_2(X_b), \ldots, g_p(X_b), X_f], \qquad (1)$$

if and only if column multiplicity of Karnaugh map $v(X_f \mid X_b) \leq 2^p$ [1, 2]. The sets X_b and i X_f are called a bound and a free set, respectively, where $X_b \cup X_f = \{I_{n-1}, ..., I_1, I_0\}$ and $X_b \cap X_f = \phi$.

The above theorem by Ashenhurst and Curtis is a mathematical model of technology mapping of a function $f: B^n \to B^m$ in two LUT blocks: LUT_$card(X_b)/p$ and LUT_$card(X_f)+p/m$, where LUT_a/b is the block that is a-input and b-output (Fig. 1).

The partition of a circuit is connected with the partition of function variables. One part of n-input variables is the set of variables for p-bound functions carried out in a bound block. The rest of the variables create a free set. From technology mapping of multi-output function point of view, it is key for LUTk/x blocks that card(Xb) $\leq$ k and card(Xf)+p $\leq$ k, where k indicates the number of inputs of LUT block.

In fact, due to a substantial number of variables it is necessary to use more complex models of decomposition such as iterative and multiple decomposition [2, 37]. In the case of complex models of decomposition, it is important to choose an appropriate decomposition path first, and then to gradually limit the number of variables until the number of variables of sub-circuits is lower than or equal to k.

The way of function representation has a significant influence on the process of searching for an appropriate decomposition. Representation in the form of a BDD diagram turns out to be especially vital while searching for a complex decomposition. Function decomposition is a horizontal cutting of a diagram in which the extract above the cutting line complies with a bound set, and the extract below the cutting line complies with a free set. The number of bound functions conforms the number of bits necessary to distinguish the so-called cut nodes, i.e. the nodes below the cutting line, to which the edges from the top part of the diagram are led [37–39].

Let us consider decomposition of the function $f: B^7 \to B^2$ shown in Fig. 2. The MTBDD diagram, presented on Fig. 2a, was divided with a red cutting line into two parts. The extract above the red cutting line, includes three variables E0 = {x0, x1, x2}. The edges, coming out of this extract, indicate two nodes (k, l) placed below the cutting line. In order to distinguish them, a single bit can be enough to lead to a single bound function. Multiple cutting method of BDD diagram is a kind of an expansion of the single cutting method. The extract of a diagram between red and black lines including the variables E1 = {x3, x4, x5, x6} has two roots named l and k. In order to indicate the number of necessary bound functions connected with this extract, there is a need to create a root table (Fig. 2c) [37]. The rows of a root table correspond to the roots of the analyzed extract and the columns correspond to separate paths in a diagram. A single row in a root table (associated with a given

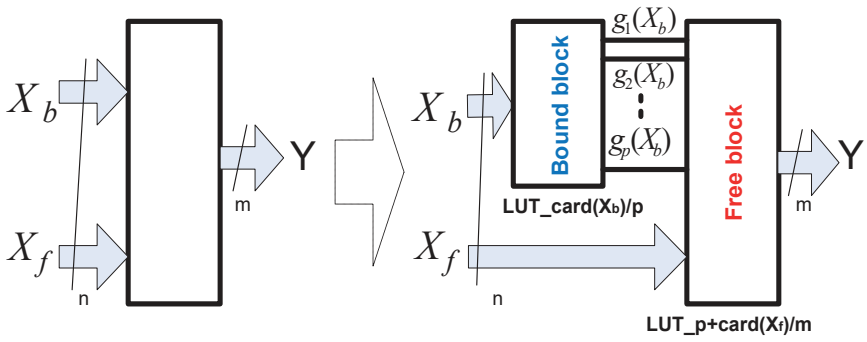

Fig. 1. Technology mapping of a function in LUT blocks that is the result of serial decomposition

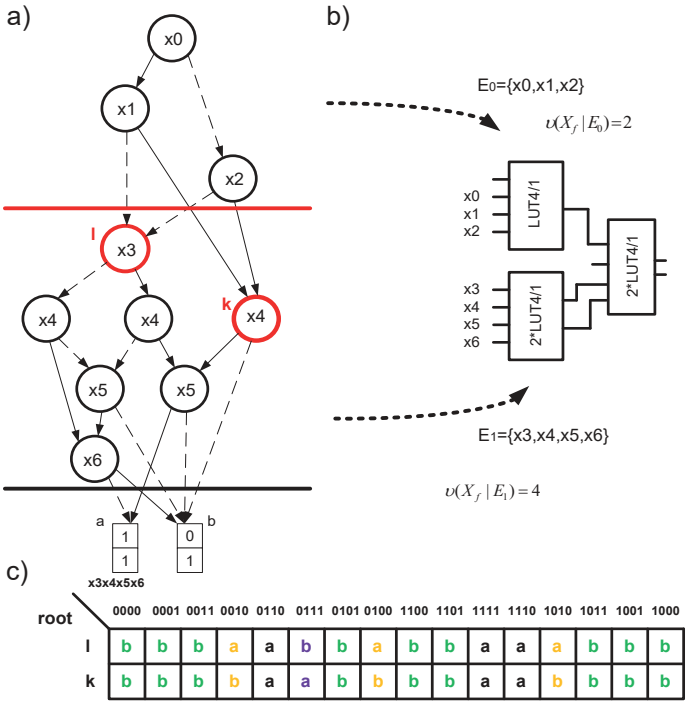

Fig. 2. Function decomposition described in the form of BDD that is the result of multiple cutting

root) will be called the vector of cut nodes. In the cells of a root table the symbols corresponding to cut nodes below the cutting line were placed (in the analyzed example, they are terminal nodes 11 and 01). The number of bound functions depends on column multiplicity of a root table υ(Xf|E1), therefore the number of various column patterns [37]. In a root table in Fig. 2 there are four column patterns, what may lead to the necessity of introducing two bound functions. Technology mapping in LUT_4/1 blocks is presented in Fig. 2b.

The essence of the paper is to present chosen synthesis strategies dedicated to LUT-based FPGA structures. Innovation of the algorithms lies in the methods of multiple cutting of a BDD diagram. The way of the cutting matches the resources of logic structures and plays a key role in the process of technology mapping of multi-output function to configurability of logic blocks.

3. Logic synthesis oriented to LUT-based FPGA

In this section the way of technology mapping of multi-output function will be presented, as it is considered to be the basis of an efficient algorithm of carrying out multi-output function in LUT-based FPGA structures. Analyzed diagrams, such as SMTBDD (shared multi-terminal binary decision diagram) presented in [37, 38], were used. In general, the SMTBDD diagram is a multi-root diagram including two multi-bit roots. The process of technology mapping, which is the basis of the proposed method, is based on appropriate multi-cutting of the MTBDD diagram.

3.1. Decomposition of a multi-output function. Advantages resulting from the usage of decomposition of multi-output functions in the process of logic synthesis include the opportunity of sharing of logic blocks. The idea of sharing a bound block is shown in Fig. 3.

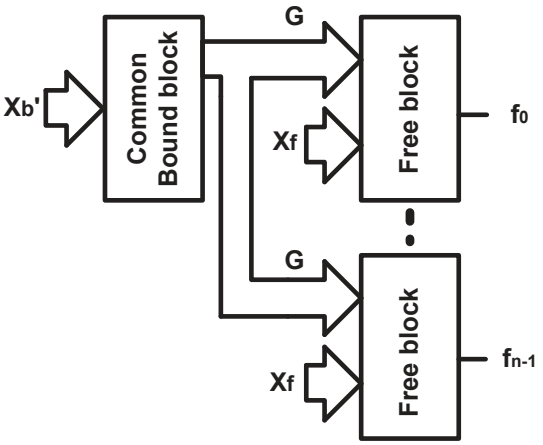

Fig. 3. The idea of sharing a bound block that is the result of decomposition of a multi-output function

In accordance with (1), G indicates the set of p-bound functions $G = \{g_1(X_b), g_2(X_b), \ldots, g_p(X_b)\}$. There is a possibility of searching for decomposition of a multi-output function through the analysis of ROBDD describing separate multi-output functions. In this case, in the process of searching for decomposition, it is necessary for the cutting line to be at the same level. In addition, separate diagrams should have the same ordering of variables above the cut lines.

The extracts of diagrams above the cutting line are associated with appropriate vectors of cut nodes. Because of the fact that the process of cutting should be done on the same level in all the diagrams, the vectors of cut nodes should have the same number of elements. In order to define the number of needed bound functions, it is necessary to introduce the notion of a partition table.

Definition 1: A partition table is a two-dimensional table whose columns are connected with appropriate paths of BDD diagram, the rows correspond with separate multi-output function, and the elements of a table include the symbols associated with cut nodes.

Column multiplicity of a partition table defines the number of necessary bound functions (card(G)) in accordance with the relation $card(G) = \lceil \lg_2 (\text{column multiplicity of a partition table}) \rceil$.

The idea of sharing a bound block in the process of decomposition with the use of BDD is presented in Example 1.

Example 1. Let us consider carrying out of three single-output functions: f0, f1, and f2 on LUT_3/1 blocks. Functions f0, f1, and f2 describe the diagrams shown in Fig. 4a. For separate

diagrams, the cutting line was led on the same level. Bound sets are three-element sets. There are three cut nodes: 0, 1, and a in the diagram connected with the function f0. In order to distinguish them, it is necessary to use two bits (two bound functions). The diagram, associated with the function f1, has two cut nodes: 1 and b for a given cutting line. In order to differentiate them, a single bit is necessary (a single bound function). In the last diagram, connected with the function f2, there are two cut nodes: c and d. Again, one bit is needed in order to distinguish them. To sum up, in the case of carrying out the functions f0, f1, and f2 separately, it is a must to use four bound functions connected with four bound blocks.

It is also possible to share bound blocks. First, the vectors of cut nodes for each of three functions (f0, f1, and f2) should be determined. Second, a two-dimensional partition table should be created (Fig. 4b). Because of the fact that the multi-output function consists of three functions, a partition table in Fig. 4b has three rows. In the partition table there are the symbols connected with appropriate cut nodes for each function. In Fig. 4b, four column patterns – A, B, C, and D – may be distinguished. The column patterns in this table correspond, respectively, to cut nodes in the MTBDD diagram that represents the multi-output function (Fig. 4c). Because of the fact that the number of the patterns (cut nodes of MTBDD diagram) is 4, two bits are needed to distinguish them. It may lead to the creation of two bound functions connected with two bound blocks (LUT3/1). Thanks to the usage of sharing of bound blocks, it was possible to reduce the number of bound functions and the number of LUT blocks. The obtained circuit structure is presented in Fig. 4d.

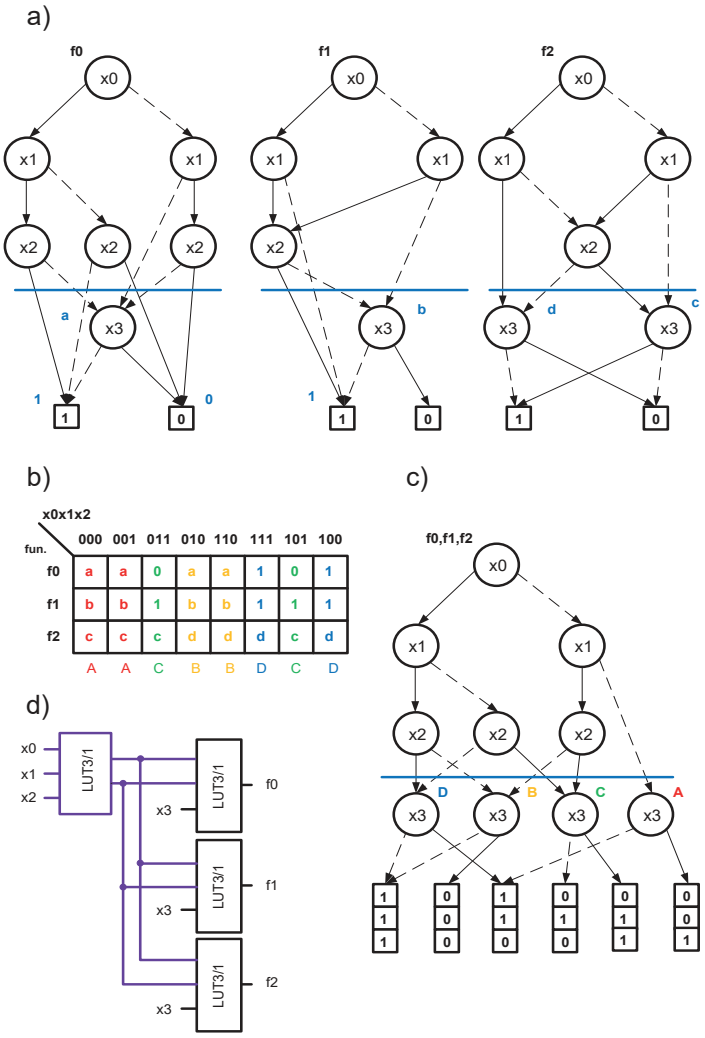

Fig. 4. Sharing resources for Example 1; a) ROBDD diagrams describing separate functions, b) a partition table, c) MTBDD for multi-output function, d) the obtained structure

The same process of searching for shared blocks may be conducted for multi-root SMTBDDs [37]. The way of proceeding turns out to be also nearly the same, except for the fact that the symbols, placed in the cells of a partition table, are connected with an appropriate column pattern in a root table and not with the cut nodes.

Unfortunately, combining too many functions into a multi-output function carried out on common resources may lead to a substantial increase in the number of bound functions and the number of LUT blocks. [40] presents the algorithm of gradual merging of BDD diagrams representing separate functions. The process of merging two diagrams is acceptable when the number of cut nodes in MTBDD diagram, gained after merging, is not higher than the number of cut nodes in the diagrams before merging. An appropriate multi-output function, carried out on shared blocks, is obtained.

In order to avoid the problem of the number of bound functions being too high, sharing of only some bound functions may be used. The idea of such sharing is presented in Fig. 5.

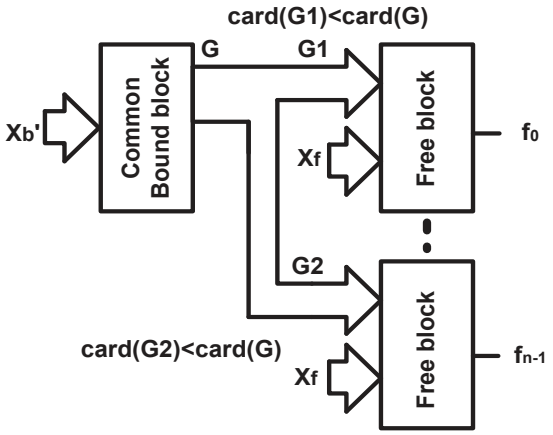

Fig. 5. The idea of sharing only some bound functions

The essence of such a solution is the appropriate coding of cut nodes in which only one code is always ascribed to a single node (unicoding). The process of searching for common bound function is a two-stage process. In the first stage, it is necessary to determine the columns of a partition table to which the same codes should be ascribed. In the second stage, separate cut nodes should be named using appropriate codes. It is essential

to determine the paths that should be named with the same code. The problem may be solved by creating equivalence classes and analyzing the consistency in a graph's structure [41, 42]. The process of searching for equivalence classes in one-root graphs is as described in [20, 42].

It is possible to determine equivalence classes in SMTBDD diagrams on the basis of the analysis of a consistency graph connected with a root table.

Example 2. Let me consider two functions described using the diagrams shown in Fig. 6a and 6b. The cutting of a diagram was done on the same levels. The extracts, which are the result of cutting (Fig. 6c and 6d), are associated with a bound set that includes the variables x2 and x3 for the function f0, and x2, x3, and x4 for the function f1. Both SMTBDD diagrams have two roots, and thus the root tables corresponding to the diagrams have two rows (Fig. 6e and 6f). Column multiplicity of tables is 4. In order to distinguish column patterns, two bits are needed, what leads to the creation of two bound functions.

In order to find a common coding bit, a consistency graph of a root table shall be created, whose nodes will be connected

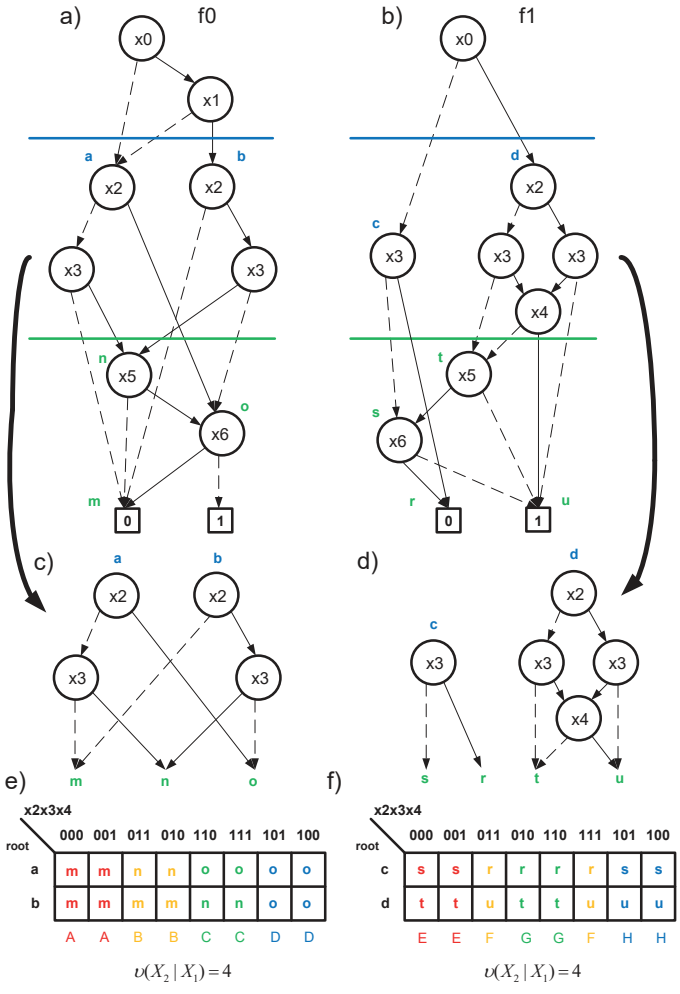

Fig. 6. The functions for which equivalence classes are searched: a, b) ROBDD diagrams, c, d) SMTBDD diagrams placed between the cutting lines, e, f) root tables associated with SMTBDD diagrams

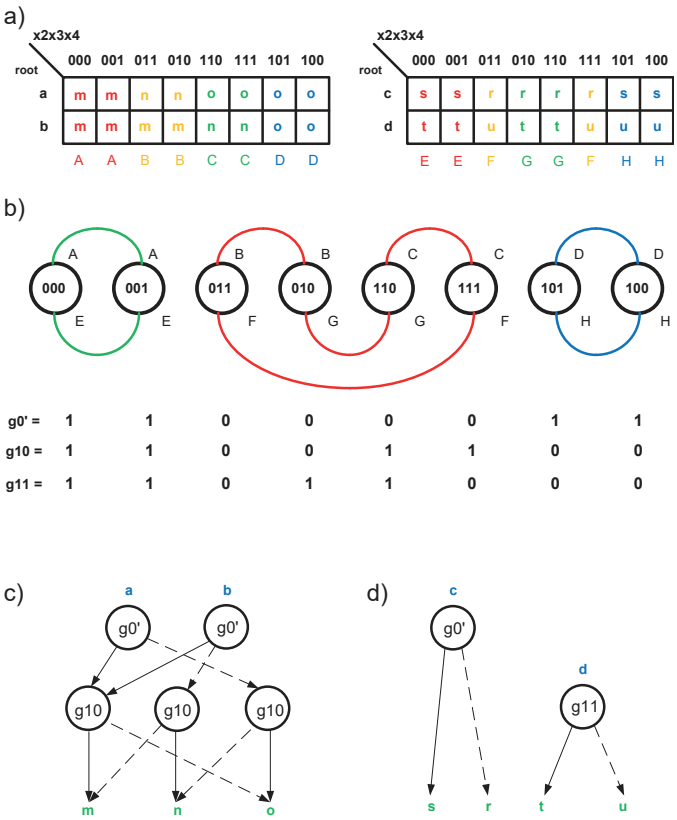

Fig. 7. Equivalence classes in SMTBDD: a) root tables associated with appropriate SMTBDD diagrams, b) consistency graph together with the values of bound functions, c, d) SMTBDD created after placing the nodes connected with bound functions g (g0 – common function)

with appropriate combinations of variables in the analyzed extract (Fig. 7b). Each combination (path) is associated with an appropriate column pattern of a root table. It becomes possible to combine the nodes (connected with the same column patterns) with edges. In the obtained consistency graph of column patterns, there may be distinguished the sets that have the edges combined with each other, marked with appropriate colors in Fig. 7b. The sets of consistency graph nodes correspond to equivalence classes. Therefore, there are three equivalence classes. Common bound function g0 enables differentiating them in the way that the value of 0 corresponds to equivalence class connected with the set of nodes marked with red color, and the value of 1 corresponds to the rest of the nodes. Basic SMTBDD diagrams in Fig. 6c and 6d may be replaced with the diagrams including the nodes associated with bound functions g0, g10, and g11 (Fig. 7c, 7d).

After having placed new SMTBDD diagrams between the cutting lines, the ROBDD diagrams, presented in Fig. 8a and 8b, are obtained. It can be observed that in both diagrams there is a node connected with a common bound function g0'. The obtained technology mapping is shown in Fig. 8c, and common logic resources are marked with a red color.

In the analyzed case both root tables have the same number of rows. However, it is not so important as in the process of cre-

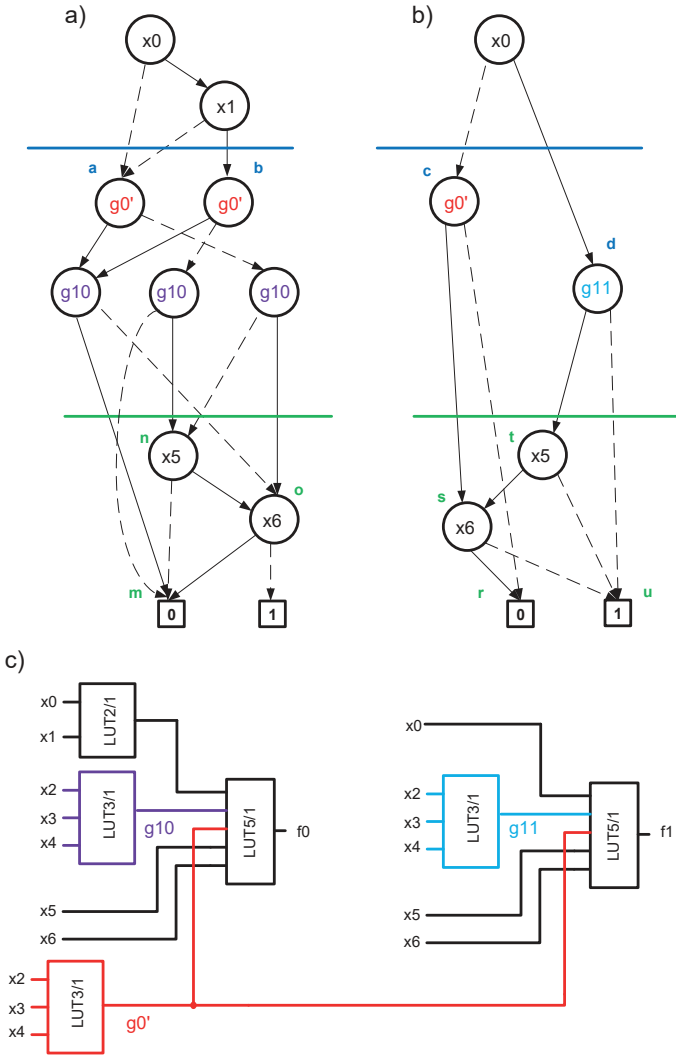

Fig. 8. The results of decomposition: a, b) the diagrams obtained for common bound functions, c) technology mapping in LUT_x/1 blocks

ating a consistency graph, it is necessary to know which pattern should be chosen for which combination and not what it is like.

3.2. Non-disjoint decomposition. In the process of decomposition, various models of non-disjoint decomposition, for which Xb ∩ Xf = Φ, are used. It often turns out that non-disjoint decomposition leads to better results as far as the area is concerned. Non-disjoint decomposition reduces the area in bound blocks.

In general, non-disjoint decomposition is a kind of an expansion of serial (disjoint) decomposition. As opposed to the latter, one part of variables may be included in both a bound set and a free set. Thus, the third set of variables, Xs, including common variables (2), may be distinguished.

$$Xb \cap Xf \neq \Phi; \qquad Xb \cap Xf = Xs. \tag{2}$$

The essence of non-disjoint decomposition is presented on Fig. 9

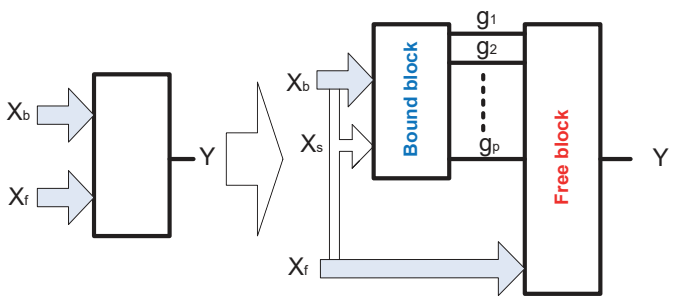

Fig. 9. The essence of non-disjoint decomposition

The process of attaching some variables to both a bound and a free set may result in the reduction of the number of bound functions g [42]. It means that the variables from the common set Xs will fulfill the role of bound functions. Not every variable can fulfill the role of a 'switching' function. Therefore, the essence of searching for non-disjoint decomposition is based on searching for variables that may fulfill the role of bound functions. Non-disjoint decomposition may significantly improve the efficiency of using logic resources [43–45]. That is why this is the main topic of many papers. The ways of searching for non-disjoint decomposition were presented in [46, 47]. The starting point of the proposed searching for non-disjoint decomposition is disjoint decomposition (e.g. a simple serial decomposition). The process of searching for appropriate decomposition includes attaching next variables to the set Xs and checking whether the attachment of a variable xi to the set Xs will lead to the reduction of the number of bound functions g. A classic method that checks this condition is based on the analysis of the vectors of cut nodes obtained for a stable value of a 'switching' variable (variable x0) and was shown in the form of Example 3.

Example 3. Let us consider the function $f: B^4 \rightarrow B$ described using ROBDD diagram presented in Fig. 10. Let us consider the cutting of a diagram on the third level counting from the root (Fig. 10a). There are three cut nodes – i, j, and k –

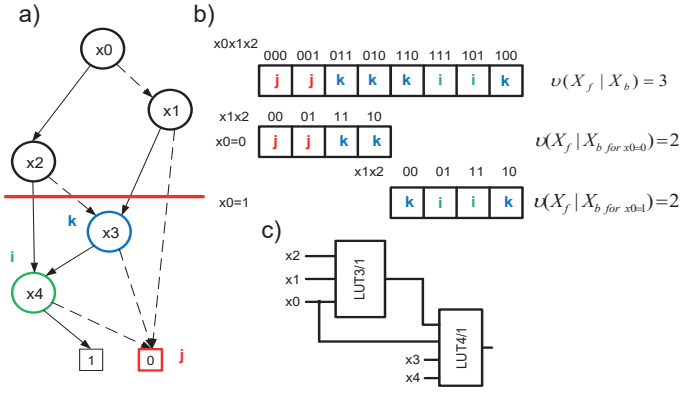

Fig. 10. Non-disjoint decomposition: a) the diagram representing the function, b) the vectors of cut nodes, c) the result of technology mapping

for the analyzed cutting. Two bits (two bound functions) are needed to differentiate them. The vector of cut nodes was presented in Fig. 10b. In order to minimize the number of bound functions, searching for a variable belonging to a bound set {x0, x1, x2}that may replace one of bound functions, should be started.

Let us establish whether the variable x0 may fulfill the role of bound function. Fig. 10b presents the partition of a vector of cut nodes for the variable x0 = 0 and x0 = 1. As the result of partition, two four-element vectors were created. The first one for x0 = 0 includes the symbols associated with two cut nodes j and k, and thus, one bound function is necessary. It is similar in the case of the vector connected with x0 = 1, where there are also two symbols, i and k (a single bound function again). The fact that the number of required bound functions for x0 = 0 and x0 = 1 is lower by 1 than the number of bound functions for non-disjoint decomposition. It means that the variable x0 may fulfill the role of bound function. The variable x0 shall be attached to both a bound block and a free block. The structure of a circuit after decomposition is shown in Fig. 10c.

In the case of multi-root SMTBDD diagrams, the method of searching for non-disjoint decomposition can be defined similarly. Disjoint decomposition is the starting point of searching for non-disjoint decomposition associated with a given extract. All the variables belonging to an SMTBDD diagram are analyzed as far as the possibility of replacing bound functions is concerned. This process is based on the attachment of variables to the set Xs and checking whether it is profitable. If the attachment of the variable xi to the set Xs causes the reduction of the number of bound functions g, the variable xi will remain in the set Xs.

It is similar as in the case of the one-root diagram, where it is essential to develop a method that will decide whether the attachment of the variable xi to the set Xs reduces the number of bound functions.

Each variable xi corresponds to the nodes in the SMTBDD diagram placed on a given level. The variable xi may take the value of 0 (xi = 0) or the value of 1 (xi = 1), which is connected with appropriate edges coming from a given node. These edges indicate appropriate subdiagrams. There are subdiagrams for xi = 0 and xi = 1 that may be distinguished. All in all, both subdiagrams indicate a given number of cut nodes for given roots. There is a possibility to create root tables for xi = 0 and xi = 1, for which the column multiplicity is determined. The number of various column patterns determines the number of bits (bound functions) necessary to distinguish them for both variable values – xi = 0 and xi = 1. If the number of bits (bound functions) needed to differentiate column patterns in a root table for the nodes indicated by the subdiagram connected with xi = 0 is lower than the number of bits (bound functions) for non-disjoint decomposition, and the number of bits (bound functions) for the subdiagram connected with xi = 1 fulfills the same condition, the variable xi may fulfill the role of a bound function. The analyzed method of searching for non-disjoint decomposition for multi-root SMTBDD diagram is presented in Example 4.

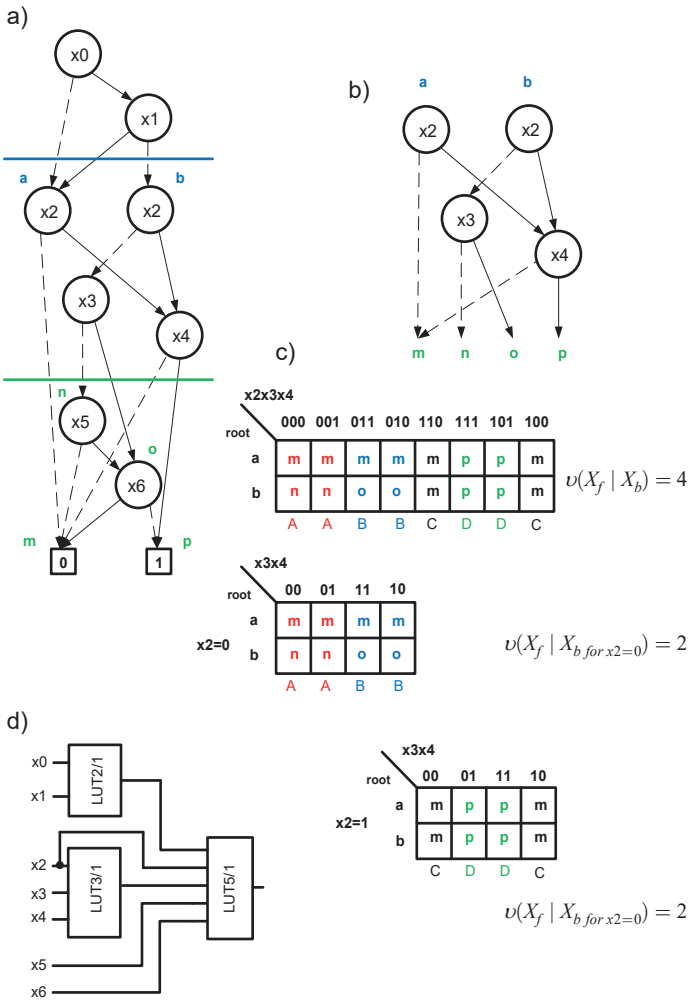

Fig. 11. Non-disjoint decomposition in SMTBDD diagrams: a) ROBDD diagram together with cutting lines, b) SMTBDD diagram, c) root tables, d) the obtained structure

Example 4. For the function described with a diagram presented in Fig. 11a, the extract between two cutting lines, including three variables E = {x2, x3, x4}, is separated. As a result of the cutting, an SMTBDD diagram with two roots, a and b is created. The SMTBDD diagram is connected with four cut nodes – m, n, o and p (Fig. 11b). In order to determine the number of bound functions, a root table is created (Fig. 11c). The column multiplicity of a root table is 4, which is why it is necessary to create two bound functions.

To replace one of them with the variable x, searching for non-disjoint decomposition should be started. First, it is considered to use the variable x2 as a 'switching' variable. Fig. 11c shows two root tables associated with x2 = 0 and x2 = 1, respectively. In both cases the column multiplicity is 2. Thus, one single bit will be enough to differentiate them. The number of bound functions is lower than the number of bound functions for non-disjoint decomposition for both x2 = 0 and x2 = 1, which means that the variable x2 may fulfill the role of a bound function. In the analyzed case, only one single variable x capable of fulfilling the role of the function g can exist.

Therefore, the searching for non-disjoint decomposition should be finished. The obtained structure of a circle is presented in Fig. 11d.

3.3. Technology mapping oriented to configurability of logic blocks. The essence of an efficient synthesis is to guarantee the best technology mapping to logic blocks included in FPGA structures. Modern logic blocks such as ALM blocks (adaptive logic module) [48], include several LUT blocks inside. There is a possibility to configure ALM blocks in various ways. Separate configurations differ in the number of inputs of LUT blocks taken into account. The number of ALM block configurations is limited. It would be especially advantageous to direct decomposition at technology mapping of a function to a concrete ALM block configuration. Thus, it is key to find such decomposition that will guarantee the possibility of technology mapping in those LUT blocks which have a given number of inputs. In the case of function representation in the form of BDD, it can be obtained by carrying out multiple decompositions [37, 38] with given cutting levels.

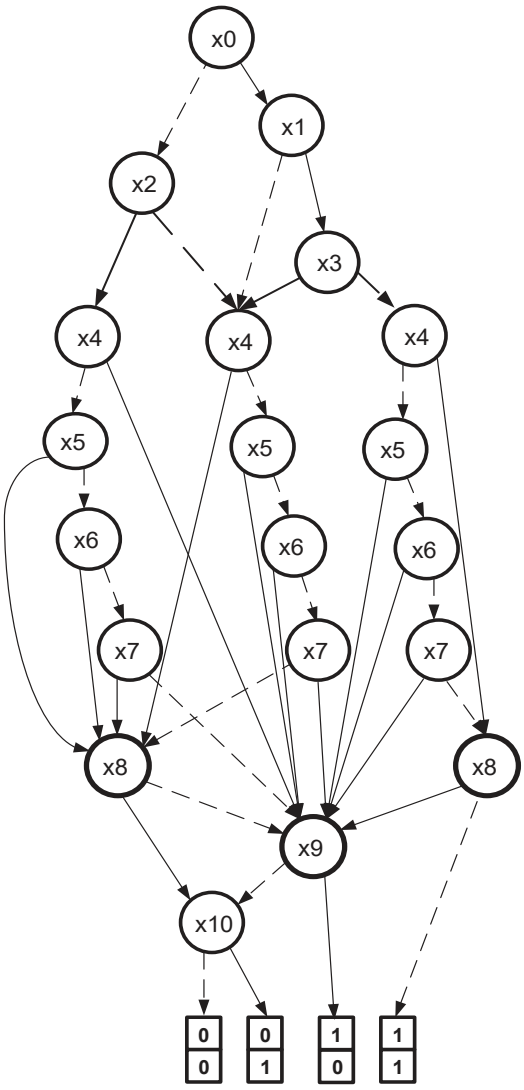

Fig. 12. MTBDD diagram representing multi-output function

Example 5. Let us consider two multi-output functions, described using the MTBDD diagram presented in Fig. 12. In order to guarantee the best technology mapping to ALM blocks, the process of decomposition should be matched to possible configurations of ALM blocks [48] and should choose the configuration that will be able to guarantee the most effective technology mapping of the analyzed multi-output function.

From the point of view of the analyzed methods of multiple cutting of BDD, the configurations in which there is no sharing of the inputs of LUT blocks are the most efficient ones. Searching for effective carrying out of a bound block should be performed best with the use of the model of multiple decomposition [2]. LUT blocks included in the ALM block may be associated with appropriate extracts of a diagram. The ALM block configuration that enables the best mapping of the results of multiple decomposition to logic resources of a circuit is shown in Fig. 13.

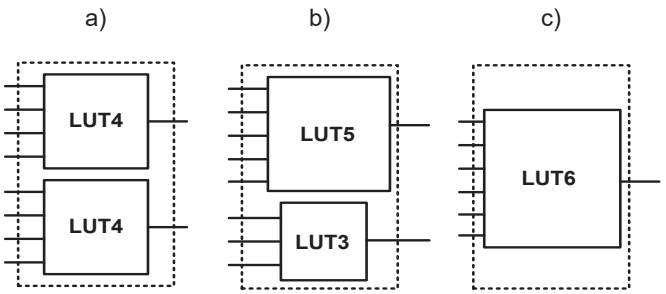

Fig. 13. ALM block configurations which guarantees the best technology mapping in the case of multiple decomposition

The number of inputs of separate LUT cells included in ALM blocks determine the cutting levels of the diagram in Fig.12. The key problem is to determine whether it is possible to conduct decomposition that will be able to match a circuit to a single ALM block. Therefore, it becomes necessary to analyze the cases for the cuttings associated with the configuration presented in Fig. 13a (on levels 4 and 8, counting from the root), Fig. 13b (on levels 5 and 8, or 3 and 8, counting from the root), and Fig. 13c (on level 6, counting from the root). It should be noticed that the configuration of the ALM block shown in Fig. 13c enables to carry out only one extract of a diagram.

The thing that should be considered first is the cutting of a diagram enabling to carry out separate extracts in an ALM block configured as shown in Fig.13a. Thus, the cutting of MTBDD diagram on levels 4 and 8, counting from the root, is analyzed and presented in Fig. 14a. For a zero extract, for which the set E0 = {x0, x1, x2, x3}, two bound functions are necessary to distinguish three cut nodes marked in blue. As the result of searching for non-disjoint decomposition, it can be said that the variable x0 may fulfill the role of a switching function.

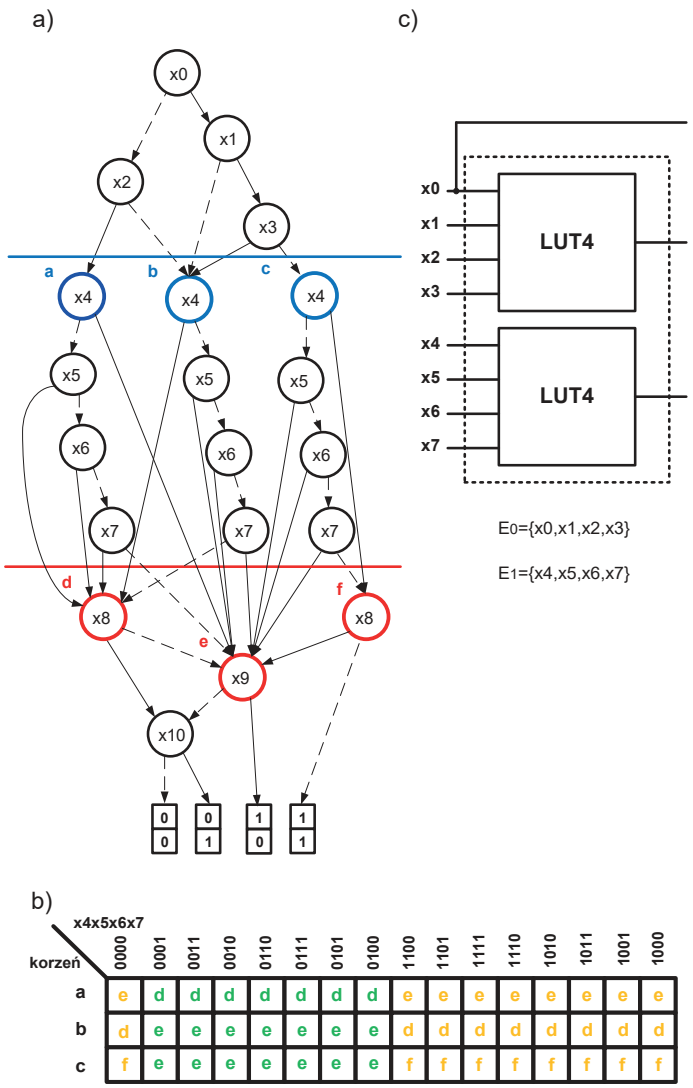

Fig. 14. Decomposition carried out by multiple cutting: a) MTBDD diagram subjected to multiple cutting, b) a root table associated with the extract between cutting lines, c) the obtained structure corresponding to bound blocks inside ALM block

In this case, one of the bound functions will be carried out in the LUT4/1 block and the second one will be replaced with the variable x0. Searching for technology mapping of the first extract, for which E1 = {x4, x5, x6, x7}, requires an analysis of the root table shown in Fig. 14b.

Column multiplicity of the root table is 2. Thus, only a single bound function is needed. It means that a bound block connected with the extract 1 may be carried out in one LUT4/1 block included in the ALM block.

As a result of a double cutting of the MTBDD diagram, the structure presented in Fig. 14c is created and carried out in a single ALM block.

There is a possibility to configure ALM blocks using other ways, and thus, it is necessary to consider other possible technology mappings. While analyzing the configuration shown in Fig. 13b, two other ways of MTBDD diagram cutting shall be considered. First is the one in which the cutting lines are led on levels 3 and 8, counting from the root (Fig. 15a), and the second, in which the cutting lines are led on levels 5 and 8 (Fig. 16a).

In the first case, together with the cutting lines presented in Fig. 15a, the zero extract (E0 = {x0, x1, x2}) has only one root. Thus, the number of bound functions depends on the number of cut nodes. Differentiation of three cut nodes leads to a technology mapping in which two bound functions exist. As in the previous case, the variable x0 may fulfill the role of a bound function. Therefore, in order to carry out a bound block associated with this extract, an LUT3/1 block is enough and it is the element of a complex configuration of the ALM block (Fig. 13b). For the first extract (E1 = {x3, x4, x5, x6, x7})

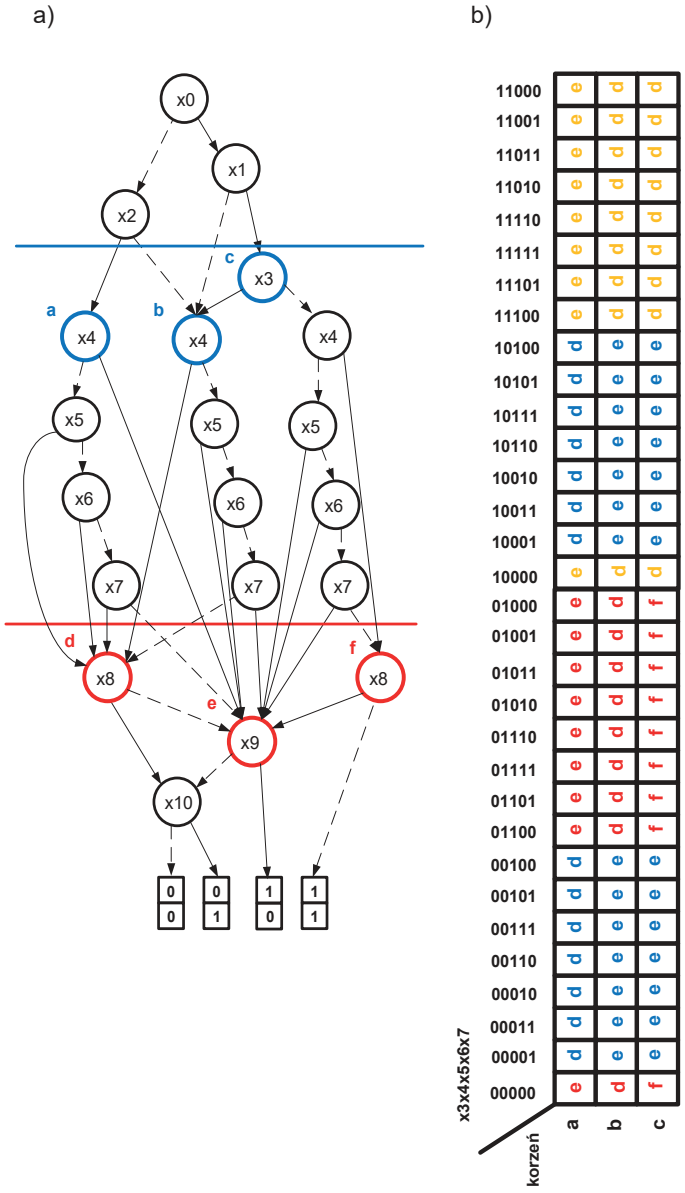

Fig. 15. Cutting of an MTBDD diagram: a) the diagram that has undergone multiple cutting on levels 3 and 8, b) the root table associated with the extract between cutting lines

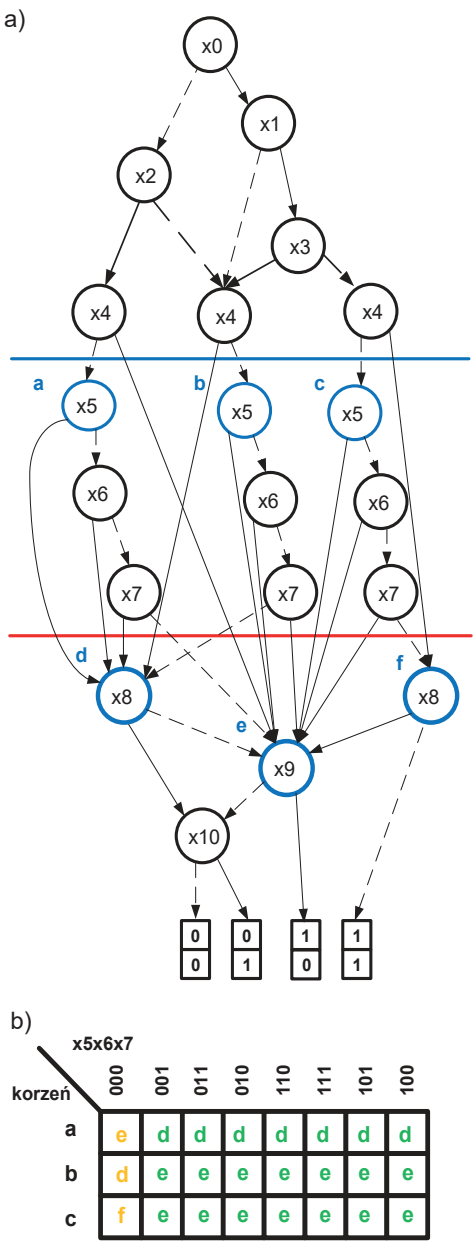

Fig. 16. Cutting of an MTBDD diagram: a) the diagram that has undergone multiple cutting on levels 5 and 8, b) the root table associated with the extract between cutting lines

column multiplicity should be determined on the basis of the root table shown in Fig. 15b. Column multiplicity of this table is 3, and thus, two bound functions are needed. It requires to use two LUT5/1 blocks. In the ALM cell for the configuration shown in Fig. 13b, only one LUT block exists. Therefore, for a complex cutting of MTBDD diagram, there is no possibility to carry out bound blocks in a single ALM block. In this case, it is necessary to use two ALM blocks. That is why it is worse than the solution matched to the configuration shown in Fig. 13a.

An alternative method of cutting the MTBDD diagram is presented in Fig. 16a. The extract, associated with the set E0 = {x0, x1, x2, x3, x4}, is identified with the edges coming out from the nodes included in it, indicating that there are six cut nodes. It requires to carry out a bound block in the form of a structure with three outputs (three bound functions). For such cutting lines, no variable included in extract 0 can fulfill the role of a switching function that would create the possibility to conduct non-disjoint decomposition. Thus, it is necessary to use three LUT5/1 blocks. Extract 1 (E1 = {x5, x6, x7}) is connected with a root table shown in Fig. 16b, whose column multiplicity is 2. In order to carry out a bound block associated with it, an LUT3/1 block is enough. The obtained solution is much further from ideal than the solutions discovered in the previously analyzed cases.

The last configuration of the ALM block which shall be considered is the configuration in which the ALM block is included in an LUT6/1 block (Fig. 13c). In this situation, the cutting of a diagram should be done on level 6 (Fig. 17). The number of cut nodes (marked in blue) in the diagram (Fig. 17) is 6. Therefore, it is necessary to use three LUT6/1 blocks, and what is more, three ALM blocks. This kind of solution is also worse than the previous ones.

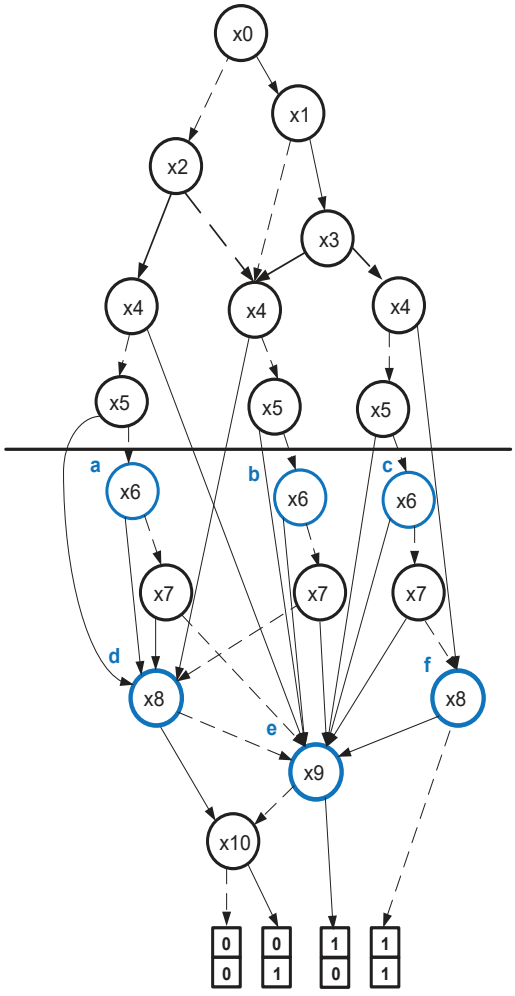

Fig. 17. The MTBDD diagram that has undergone multiple cutting on level 6

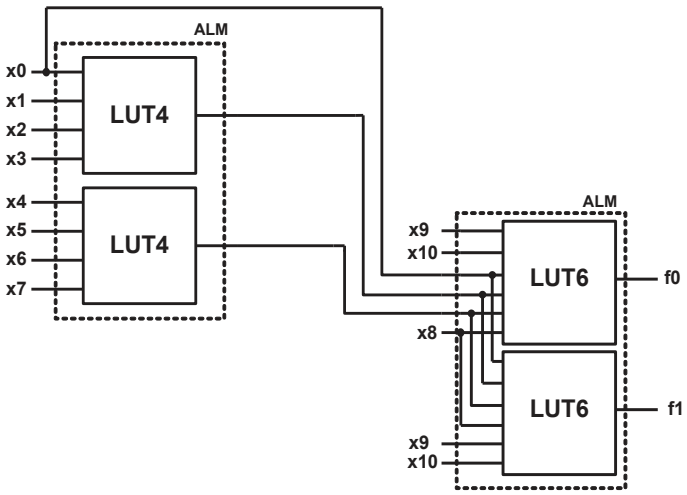

Fig. 18. The results of the synthesis – technology mapping of multi-output function in ALM blocks

Among the analyzed cuttings of a diagram, matched to configurable abilities of ALM blocks, the best results were obtained in the configuration of two independent LUT4/1 blocks. In this situation, bound blocks can be carried out in a single ALM block. After having replaced the extracts 0 and 1 with three bound functions (one of them is the variable x0), the diagram describing the free block, analyzed in the next stage of synthesis, includes six variables. It results from the fact that both bound functions included in a multi-output function and describing a free block depend on the same six variables. Thus, these blocks may be carried out in a single ALM block configured in such a way that part of the inputs are shared (4 out of 6) for both LUT6/1 blocks [48].

As a result of the synthesis of the analyzed multi-output function with the use of multiple cutting method of MTBDD diagram directed at technology mapping to ALM blocks, the obtained structure consists of two ALM blocks and is presented in Fig. 18.

It should be noted that in the analyzed example the considerations are based on only one ordering of variables. In the analyzed synthesis system, called MultiDec, the search is for the best mapping, such that would use the lowest number of blocks, taking into consideration the configuration of ALM blocks.

4. Experimental results

The base methods of decomposition are implemented in the MultiDec algorithm that is described in detail in [37].

Additionally, the logic synthesis strategy directed to configurability of logic blocks with the use of the MultiDec method also includes:

a. initial grouping of functions in multi-output function
b. the choice of cutting lines being oriented towards the configurability of logic blocks
c. searching for disjoint decompositions for separate multi-output functions (various ordering of variables)
d. optimization by searching for non-disjoint decomposition
e. searching for shared bound functions (unicoding)
f. checking if there is a necessity to carry out next stages of decomposition

The matter of variable ordering in a BDD diagram is essential from the decomposition point of view. The number of cut nodes depends on (apart from the level on which the cutting was done) the BDD ordering. In the classic approach to decomposition with the use of BDD, in which there is only one single horizontal cutting line, the problem of ordering is based on determining which variables should be above and below the cutting line. In the case of multiple cutting (MultiDec), it is key to determine in which extract (between which cutting lines) the given variable is. In the case of MultiDec, three main stages of searching for effective variable ordering may be determined:

1. Initial variable ordering on the basis of statistical frequency of occurring of a given variable in the description of the analyzed multi-output function
2. The levels of cutting lines
3. Relocation of a given variable between separate extracts (bound sets) – each time the assessment of mapping technology to a given blocks is performed (after the analysis of all the extracts, a variable is placed in that extract in which it is the most profitable – it results from the analysis of mapping the efficiency cofactor)

Stage 3 is carried out for all the variables in BDD. Searching for the best possible variable ordering in MultiDec does not guarantee that the optimal solution will be found, since not all variable orderings in BDD are analyzed. However, it enables to reduce the time of searching for an acceptable ordering of variables in BDD which is longer than in the classic approach with one cutting line.

In order to determine the efficiency of the proposed solutions, it is necessary to compare the results of synthesis using the MultiDec method with the results obtained with the use of academic, as well as commercial tools.

It is difficult to compare the MultiDec method with academic tools. A substantial majority of these tools is oriented to technology mapping of a function to homogenous LUT blocks that have a given number of inputs. At present, ABC [31] is an academic system that enables to carry out the synthesis in both combinational and sequential circuits. The comparison of the MultiDec and ABC systems is performed. The ABC system is characterized by flexibility, and its synthesis results depend on the scripts including the set of synthesis instructions. The MultiDec system is compared with three various scripts, such as ABC_1, ABC_2, ABC_3 and ABC_4. The scripts ABC_1 (strash; dch; if -K 5; mfs) and ABC_3 (strash; dch; if -K 6; mfs) enable to conduct technology mapping in logic blocks that have 5 and 6 inputs, respectively. The script ABC_2 (strash; resyn2; fpga), on the other hand, enables to obtain results that are the outcome of an advanced resynthesis together with matching to logic blocks with 5 inputs. In addition, the experiment for the script ABC_4 (&st; &synch2; &if -K 5;) was conducted using the package ABC9. A series of experiments for a popular set of benchmarks are conducted [49].

Table 1
The comparison of the MultiDec system and the ABC system

Benchmarks			MultiDec_45			MultiDec_56			ABC_1 (strash; dch; if -K 5; mfs)			ABC_2 (strash; resyn2; fpga)			ABC_3 (strash; dch; if -K 6; mfs)			ABC_4 (&st; &synch2; &if -K 5;)		
Name	Inputs	Outputs	LUT_45	Levels	T [ms]	LUT_56	Levels	T [ms]	LUT_5	Levels	T [ms]	LUT_5	Levels	T [ms]	LUT_6	Levels	T [ms]	LUT_5	Levels	T [ms]
5xp1	7	10	14	2	265	8	2	296	23	3	500	25	3	1110	15	2	480	18	2	390
b12	15	9	22,5	3	561	11,5	2	592	19	3	280	17	2	880	14	2	280	19	2	110
cm163a	16	5	6	3	670	5,5	2	452	9	2	220	9	2	830	7	2	220	9	2	160
cm85a	11	3	15	3	483	5	2	546	8	2	220	10	3	790	9	2	220	11	2	110
con1	7	2	2,5	2	46	1,5	1	15	3	2	170	3	2	780	2	1	170	3	2	130
f51m	8	8	15,5	4	249	5,5	2	156	26	3	260	34	3	910	13	3	270	24	3	140
inc	7	9	21	3	312	9	2	109	27	3	270	25	3	960	13	2	260	27	3	120
misex1	8	7	10,5	3	265	6,5	2	156	16	2	230	19	2	880	9	2	230	16	2	130
pcle	19	9	11,5	3	2340	9	3	3525	15	2	230	15	2	980	12	2	230	14	2	140
rd73	7	3	4,5	2	124	3	2	140	17	3	250	20	4	820	14	3	230	18	3	160
rd84	8	4	8	2	280	6	2	680	45	4	390	58	4	970	34	3	310	60	4	160
sqn	7	3	10,5	3	140	5	2	140	22	3	270	25	3	770	11	2	270	19	3	110
sqr6	6	11	12	2	140	6	1	31	21	3	310	24	3	970	11	1	250	20	2	140
sqrt8	8	4	6	2	124	4,5	2	78	11	3	280	12	3	840	9	3	230	12	3	110
t481	16	1	3,5	3	374	4,5	4	468	13	4	270	18	4	850	15	3	230	20	4	110
x2	10	7	12,5	3	280	6	2	680	13	2	250	14	2	860	12	2	200	14	2	130
z4m1	7	4	5,5	3	124	6	2	124	5	2	200	7	2	830	5	2	200	5	2	110
x5xp1	7	10	14	2	234	8	2	202	20	3	280	19	3	930	16	2	250	19	3	140
ldd	9	18	21	2	1185	18,5	2	1482	26	2	260	35	2	920	26	2	250	27	2	110
Sum:			216	50	8196	129	39	9872	339	51	5140	389	52	16880	247	41	4780	355	48	2710

The experimental results are presented in Table 1. For the MultiDec system in the first column (MultiDec_45), technology mapping was oriented to popular configurable blocks to the form of LUT_4/2 or LUT_5/1. In the second column (MultiDec_56), technology mapping is directed to configurable blocks to the form of LUT_5/2 or LUT_6/1.

The obtained results are compared on the basis of total number of blocks and levels included in the last row in Table 1. The values are presented in the form of bar charts shown in Fig. 19a (blocks) and Fig. 19b (levels).

While comparing the obtained results as far as the number of blocks is concerned (Fig. 19a), it can be observed that it

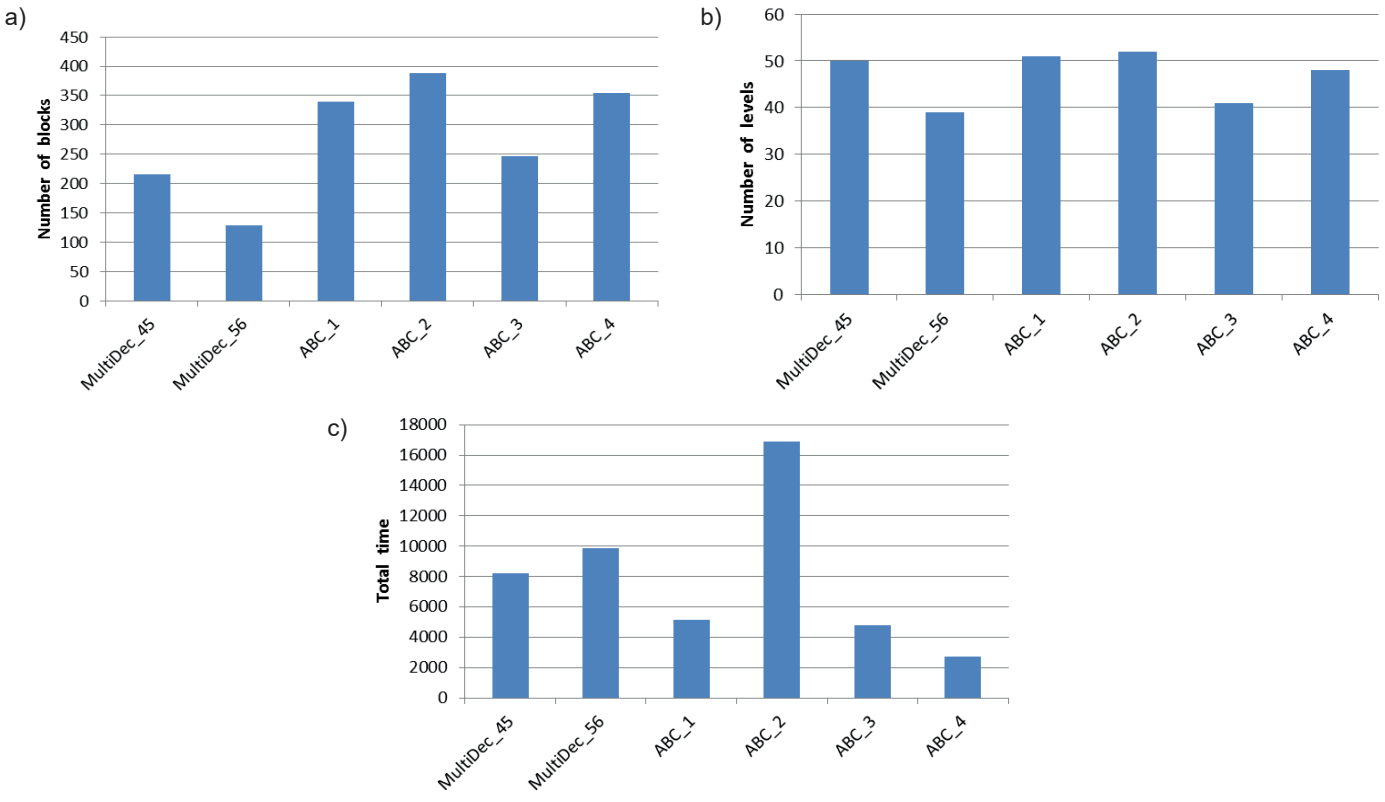

Fig. 19. The comparison of the MultiDec system and the ABC system for: a) total number of blocks, b) total number of levels, c) total synthesis time

is essential to use configurability of logic blocks. Technology mapping that enables flexible choosing of the number of inputs causes a substantial reduction in the number of necessary blocks comparing to the solutions with a precisely defined number of inputs in a logic block. While comparing the number of logic levels (Fig. 19b), it can be said that the obtained results differ from each other. After having compared the total synthesis time, it can be said that despite the lack of the resynthesis process in the MultiDec system, the synthesis process in ABC system is definitely faster.

The comparison of the MultiDec system with commercial tools is much more precise. The MultiDec system makes it possible to create a structural description of the results in HDL. Therefore, it enables to conduct the last stages of synthesis with commercial tools. A series of experiments were performed with the use of commercial tools in order to compare the effects of synthesis for the cases in which decomposition was carried out in the MultiDec system, with the solutions in which the total synthesis process was carried out with a commercial tool. The results were compared with the most popular and widely available synthesis tools such as Quartus 15.1 (Altera) and ISE 14.7 (Xilinx). In the case of Quartus, the synthesis was oriented to ALM blocks included in the circles Stratix and Cyclon series V. In the case of ISE, the synthesis was directed to the blocks included in the circles Artix, Kintex series 7, and Virtex series 6 (they are characterized by the same configurable abilities). The results of the comparison are presented in Table 2.

Table 2
The comparison of the synthesis results as far as the number of blocks for commercial tools is concerned

Benchmarks			ISE	ABC_3 + ISE	MultiDec + ISE	Quartus	ABC_3 + Quartus	MultiDec + Quarus
Name	Inputs	Outputs						
5xp1	7	10	14	14	13	11	9	8
b12	15	9	16	15	20	10	9	13
cm163a	16	5	7	7	10	5	5	6
cm85a	11	3	6	6	8	5	6	10
con1	7	2	2	2	2	1	1	1
f51m	8	8	13	14	10	8	9	6
inc	7	9	11	11	13	11	11	11
misex1	8	7	9	9	9	6	6	7
pcle	19	9	12	12	12	9	9	8
rd73	7	3	19	11	6	8	6	3
rd84	8	4	13	14	9	41	26	6
sqn	7	3	6	6	9	8	8	8
sqr6	6	11	10	10	10	7	7	7
sqrt8	8	4	11	11	6	6	8	4
t481	16	1	4	10	8	3	3	6
x2	10	7	11	11	11	7	8	10
z4m1	7	4	5	5	11	5	3	5
x5xp1	7	10	14	15	16	11	10	10
ldd	9	18	26	24	27	15	14	18
Sum:			209	207	210	177	158	147

In the second table, the first two columns describe the synthesis results (the number of blocks) for ISE. The column ISE describes the case in which the total synthesis process was carried out in the ISE system. The column ABC_3 + ISE presents the result of the synthesis performed using the ABC system and the ISE tool. The column MultiDec + ISE includes the solutions in which an HDL circuit was subjected to synthesis and was achieved as a result of a decomposition carried out with the use of MultiDec. In the next two columns, the results gained with the use of Quartus are presented. The last row of the second table determines the total number of blocks for the compiled set of benchmarks obtained in each case.

While comparing the total number of blocks gained in both cases for ISE, it can be observed that the results are nearly the same. Thus, it is difficult to talk about advantages resulting from the use of the MultiDec algorithm. In the Quartus system it was possible to reduce the number of blocks by about 17%, as a result of the decomposition methods included in the MultiDec system. It is clearly visible that the elements of matching decomposition algorithms to logic resources of the FPGA structure included in the MultiDec system improve the efficiency of the process of synthesis.

5. Conclusion

The basic problem of logic synthesis of digital circuits carried out with the use of an LUT-based FPGA structure is the matter of the partition of a designed circuit into separate, configurable logic blocks. The optimal partition of circuits is not known. Many scientists have been working to improve the synthesis process oriented to LUT-based FPGA structures.

The ideas presented in this paper seem to be competitive in reference to those that are known from literature. They are a kind of a development of the classic theory of decomposition. The essence of the development is based on the way of searching for appropriate decomposition of a multi-output function described with the use of MTBDD. The proposed optimization includes a way of performing a decomposition of a multi-output function, a method of searching for non-disjoint decomposition, and multiple cutting of an MTBDD diagram, enabling to match the decomposition process to configurability of logic blocks.

Experimental results prove the effectiveness of the presented solutions in comparison with the most popular academic and commercial tools. The next stages will include the usage of proposed decomposition methods in the process of multi-level optimization.

Acknowledgements. The investigation was supported by the Polish Ministry of Science and Higher Education; fund no. 8686/E-367/S/2014.

References

[1] R.L. Ashenhurst, “The decomposition of switching functions”, *Proceedings of an International Symposium on the Theory of Switching*, April 1957.

[2] H.A. Curtis, *The Design of Switching Circuits*, D. van Nostrand Company, Inc., Princeton, 1962.

[3] R. Murgai, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vincentelli, "Improved logic synthesis algorithms for table look up architectures", *ICCAD-91*, 564–567 (1991).
[4] P. Abouzeid, B. Babba, M. Crastes, and G. Saucier, "Input-driven partitioning methods and application to synthesis on table-lookup-based FPGAs", *IEEE Trans on CAD* 12 (7), 913–925, 1993.
[5] S.D. Brown, R.J. Francis, J. Rose, and Z.G. Vranesic, *Field Programmable Gate Arrays*, pp. 45–86, Kluwer Academic Publishers, Boston, 1993.
[6] R.K. Brayton, G.D. Hachtel, C. McMullen, and A.L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, 1984.
[7] D. Chen and J. Cong, "DAOmap: A depth-optimal area optimization mapping algorithm", *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004*, 752–759 (2004).
[8] J-D. Huang, J-Y. Jou, and W-Z. Shen, "ALTO: An iterative area/performance tradeoff algorithm for LUT-based FPGA technology mapping", *IEEE Transactions on Very Large Integration (VLSI) Systems* 8 (4), 392–400 (2000).
[9] A. Mishchenko, S. Chatterjee, and R. Brayton, "Improvements to technology mapping for LUT-based FPGAs", Proc. FPGA 2006, 41–49 (2006).
[10] A. Mishchenko, S. Chatterjee, and R.K. Brayton, "Improvements to technology mapping for LUT-based FPGAs", *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.* 26 (2), 240–253 (2007).
[11] Siddhartha and N. Kapre, "Fanout decomposition dataflow optimizations for FPGA-based sparse LU factorization", *International Conference on Field-Programmable Technology (FPT)*, 252–255 (2014).
[12] Siddhartha and N. Kapre, "Breaking sequential dependencies in FPGA-based sparse LU factorization", *FCCM '14: Proceedings of the 22nd IEEE Symposium on Field Programmable Custom Computing Machines*, 1–4 (2014).
[13] Y.Y. Liang, T.Y. Kuo, S.H. Wang, and W.K. Mak, "ALMmap: Technology mapping for FPGAs with adaptive logic modules", 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 143–148 (2010).
[14] S. Chang, M. Marek-Sadowska, and T. Hwang, "Technology mapping for TLU FPGAs based on decomposition of binary decision diagrams", *IEEE Transactions on Computer-Aided Design* 15 (10), 1226–1235 (1996).
[15] M. Kubica and D. Kania, "Area-oriented technology mapping for LUT-based logic blocks", *International Journal of Applied Mathematics and Computer Science* 27 (1), 2017, (in press).
[16] Y. Lai, K.R. Pan, and M. Pedram, "OBDD-based function decomposition: Algorithms and implementation", *IEEE Transactions on Computer-Aided Design* 15(8), 977–990 (1996).
[17] N. Vemuri, P. Kalla, and R. Tessier, "BDD-based logic synthesis for LUT-based FPGAs", *ACM Trans. Design Autom. Electron. Syst.* 7 (4), 501–525 (2002).
[18] D. Kania, "Logic decomposition for PAL-based CPLDs", *Journal of Circuits, Systems, and Computers* 24 (3), 1–27 (2015).
[19] D. Kania and J. Kulisz, "Logic synthesis for PAL-based CPLD-s based on two-stage decomposition", *The Journal of Systems and Software* 80, 1129–1141 (2007).
[20] A. Opara and D. Kania, "Decomposition-based logic synthesis for PAL-based CPLDs", *International Journal of Applied Mathematics and Computer Science* 20 (2), 367–384 (2010)
[21] G. Borowik, T. Łuba, and P. Tomaszewicz, "On memory capacity to implement logic functions", Computer Aided Systems Theory – EUROCAST 2011 Part II, pp. 343–350, eds. R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Springer-Verlag, Berlin, 2012.
[22] T. Sasao, *Memory-Based Logic Synthesis*, Springer, New York, 2011.
[23] D. Kania, "Decomposition-based synthesis and its application in PAL-oriented technology mapping", *Proceedings of the 26th Euromicro Conference*, 138–145 (2000).
[24] T. Łuba and 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* 3 (3–4), 289–300 (1995).
[25] M. Perkowski, R Malvi, S. Grygiel, M. Burns, and A. Mishchenko, "Graph coloring algorithms for fast evaluation of curtis decompositions", *36th ACM/IEEE DAC'99*, (1999).
[26] C. Yang and M. Ciesielski, "BDS: A BDD-based logic optimization system", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 21 (7), 866–876 (2002).
[27] L. Cheng, D. Chen, and M.D.F. Wong, "DDBDD: Delay-driven BDD synthesis for FPGAs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27 (7), 1203–1213 (2008).
[28] L. Cheng, D. Chen, and M.D.F. Wong, "DDBDD: Delay-driven BDD synthesis for FPGAs", *44th ACM/IEEE Design Automation Conference*, 910–915 (2007).
[29] P. Fiser and J. Schmidt, "The case for a balanced decomposition process", *Proc. of 12th Euromicro Conference on Digital Systems Design (DSD)*, 601–604 (2009).
[30] P. Fiser and J. Schmidt, "On using permutation of variables to improve the iterative power of resynthesis", *Proc. of 10th Int. Workshop on Boolean Problems (IWSBP)*, 107–114 (2012).
[31] Berkeley Logic Synthesis Group, *ABC: A System for Sequential Synthesis and Verification*, Dec. 2005, http://www.eecs.berkeley.edu/~alanmi/abc
[32] V. Manohararajah, D.P. Singh, and S.D. Brown, "Post-placement BDD-based decomposition for FPGAs", *International Conference on Field Programmable Logic and Applications 2005*, 31–38 (2005).
[33] K. Muma, D. Chen, Y. Choi, D. Dodds, M.H. Lee, and S.-B. Ko, "Combining ESOP minimization with BDD-based decomposition for improved FPGA synthesis", *Canadian Journal of Electrical and Computer Engineering* 33 (3/4), 177–182 (2008).
[34] K. Muma and S.-B. Ko, "A new logic synthesis, ExorBDS", *Canadian Conference on Electrical and Computer Engineering 2005*, 816–819 (2005).
[35] A. Opara and D. Kania, "Logic synthesis strategy based on BDD decomposition and PAL-oriented optimization", *11th International Conference of Computational Methods in Science and Engineering, ICCMSE 2015, AIP Conf. Proc.* 1702 (1), 60002.1–60002.4 (2015).
[36] T. Sasao, *FPGA Design by Generalized Functional Decomposition in Logic Synthesis and Optimization*, Kluwer Academic Publishers, Boston, 1993.
[37] M. Kubica and D. Kania, "SMTBDD: New concept of graph for function decomposition", *13th IFAC Conference on Programmable Devices and Embedded Systems*, 49–54 (2015).
[38] M. Kubica and D. Kania, "SMTBDD: New form of BDD for logic synthesis", *International Journal of Electronics and Telecommunications* 62 (1), 33–41 (2016).

[39] S. Minato, *Binary Decision Diagrams and Applications for VLSI CAD*, Kluwer Academic Publishers, 1996.
[40] M. Kubica, *Decomposition and Technology Mapping Based on BDD*, PhD thesis, Silesian University of Technology, Gliwice, 2014, http://ssuise-keit.multimedia.edu.pl/doktoraty.php, [in Polish].
[41] D. Kania, *Programmable Logic Devices*, PWN, Warszawa 2012 (in Polish)
[42] Ch. Scholl, *Functional Decomposition with Application to FPGA Synthesis*, Kluwer Academic Publisher, Boston, 2001.
[43] E. Dubrova, "A polinominal time algorithm for non-disjoint decomposition of multi-valued functions", *34th International Symposium on Multiple-Valued Logic*, 309–314 (2004).
[44] E. Dubrova, M. Teslenko, and A. Martinelli, "On relation between non-disjoint decomposition and multiple-vertex dominators", *2004 IEEE International Symposium on Circuits and Systems*, vol. IV, 493–496 (2004).
[45] E. Hrynkiewicz and S. Kołodziński, "An Ashenhurst disjoint and non-disjoint decomposition of logic functions in Reed-Muller spectral domain", *17th International Conference on Mixed Design of Integrated Circuits and Systems*, 293–296 (2010).
[46] A. Opara, *Decomposition Synthesis Methods of Combinational Circuits Using BDD*, PhD thesis, Silesian University of Technology, Gliwice, 2008, http://ssuise-keit.multimedia.edu.pl/doktoraty.php, [in Polish].
[47] S. Yamashita, H. Sawada, and A. Nagoya, "New methods to find optimal non–disjoint bi–decompositions", *Design Automation Conference 1998. Proceedings of the ASP-DAC '98*, 59–68 (1998).
[48] Altera, *Logic Array Blocks and Adaptive Logic Modules in Stratix V Devices*, 2012
[49] Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University, http://www.cbl.ncsu/edu/