BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 67, No. 5, 2019
DOI: 10.24425/bpasts.2019.130880

Technology mapping oriented to adaptive logic modules

M. KUBICA* and D. KANIA

Institute of Electronics, Silesian University of Technology, ul. Akademicka 2A, 44-100 Gliwice, Poland

Abstract. This paper presents an innovative method of technology mapping of the circuits in ALM appearing in FPGA devices by Intel. The essence of the idea is based on using triangle tables that are connected with different configurations of blocks. The innovation of the proposed method focuses on the possibility of choosing an appropriate configuration of an ALM block, which is connected with choosing an appropriate decomposition path. The effectiveness of the proposed technique of technology mapping is proved by experiments conducted on combinational and sequential circuits.

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

1. Introduction

The key element of a logic synthesis dedicated to FPGA is technology mapping. This process is directly associated with a function decomposition [14, 22]. There is a range of academic tools that allow for function decomposition [5, 11, 21, 26, 27]; there are also many well known models of decomposition [4, 11, 17, 25]. Particularly popular models include AIG partitioning [3, 9, 19, 20, 23] and methods using cutting of BDD diagrams [13, 15]. Thus, the method of representation of a logic function is extremely important. It appears that the decomposition algorithms in the literature are universal. Despite the advantages these methods, most of them do not allow us to use the specific features of certain architectures of FPGAs. Depending on the way in which they are produced, the configurabilities of logic blocks within FPGAs are slightly different. Using these features in the process of technology mapping may lead to a limitation on the number of logic blocks necessary to implement logic functions. Obviously, the obtained solutions are assigned to certain logic blocks, and thus, their universal character is strongly limited (it is hard to compare them with other academic tools). However, it seems that this methodology may be used for various FPGAs.

The main goal of this paper is to present a method of technology mapping that is associated with decomposition by cutting a BDD diagram and dedicated to the specific configurabilities of modern logic blocks. The idea of technology mapping was presented in the form of adaptive logic module (ALM) blocks by Intel [10].

Section 2 discusses the theoretical basis of decomposition and the elements of technology mapping. Section 3 presents the proposed method of technology mapping of logic functions in ALM blocks, based on multiple cuttings of a BDD diagram. We propose an innovative usage of non-disjoint decomposition that enables us to implement a function in certain configurable ALM blocks. Section 4 describes the decomposition of combinational FSM blocks in ALM. Section 5 presents experimental results conducted on a wide range of benchmarks. The paper ends with a brief conclusion.

2. Theoretical background

The partition of a combinational circuit into logic blocks, included in FPGA, is directly connected with the decomposition of a function. The theoretical basis of decomposition was defined by Ashenhurst and Curtis [2, 7]. The classical approach assumes that function decomposition is based on the partition of variables into a bound set Xb and a free set Xf. A bound set is defined as a set of variables for bound functions g, and the block implementing the p bound functions is called a bound block. A free function is implemented within a block (a free block) and its variables consist of bound functions and a free set Xf. For a disjoint decomposition, a bound and a free set do not have common variables. It is essential for decomposition to determine the number (p) of bound functions needed (i.e. the number of connections between a bound and a free block).

Let f be an n-input i m-output logic function reflecting set B^n into B^m i.e. $f\colon B^n \rightarrow B^m$, where $B = \{0, 1\}$. Function $f\colon B^n \rightarrow B^m$ may be presented as $Y = f(In-1, ..., I1, I0)$, where $Y = \{ym-1, ..., y1, y0\}$.

Function $f\colon B^n \rightarrow B^m$ is subjected to decomposition i.e.

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

if and only if there is column multiplicity of the Karnaugh map $\nu(X_f \mid X_b) \leq 2^p$ [2, 7], where $Xb \cup Xf = \{In-1, ..., I1, I0\}$ and $X_b \cap X_f = \phi$.

The partition of a circuit, which is the result of a function decomposition, is illustrated in Fig. 1a. In the case of a function representation in the form of BDD [1, 18], decomposition

*e-mail: marcin.kubica@polsl.pl

Manuscript submitted 2018-08-13, revised 2019-04-22, initially accepted for publication 2019-05-19, published in October 2019

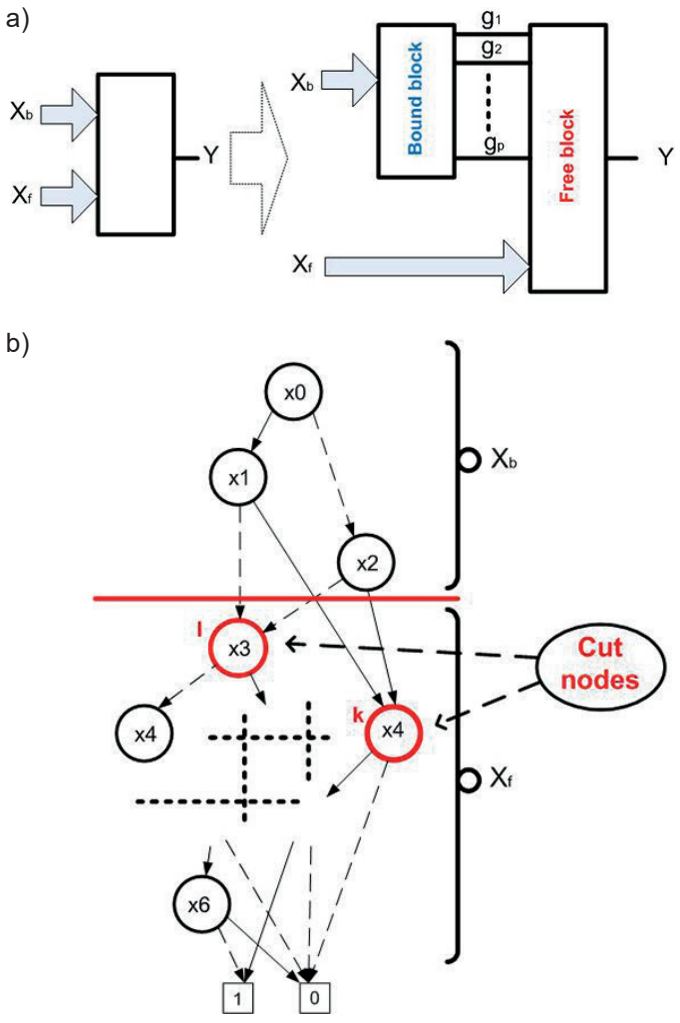

Fig. 1. Decomposition by Ashenhurst – Curtis: a) the essence of a circuit partition; b) a function decomposition resulting from a single cutting of BDD

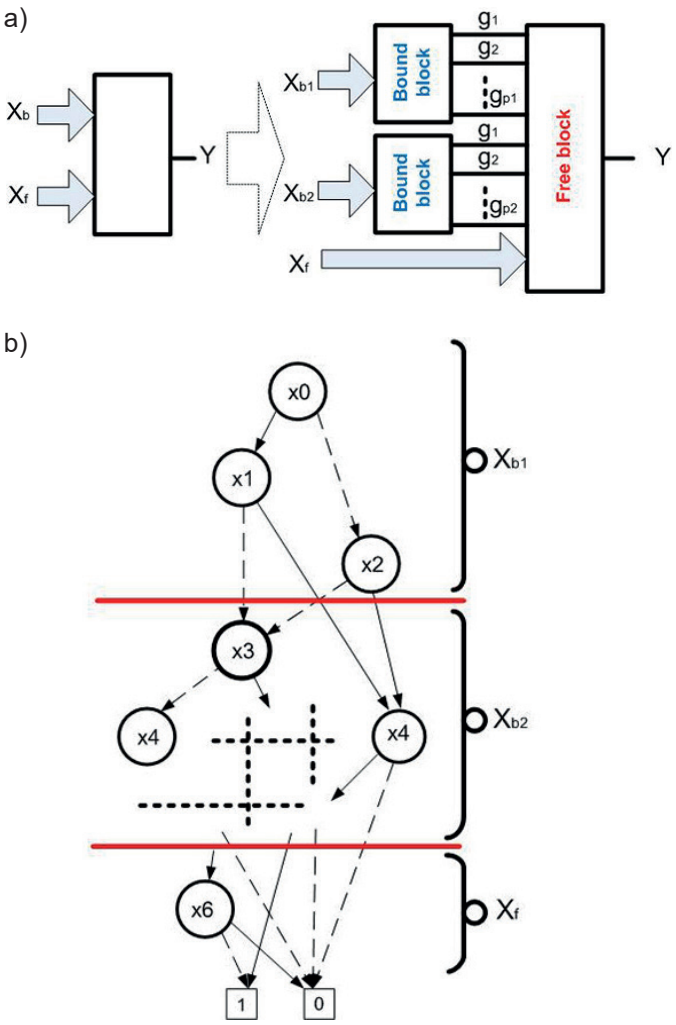

Fig. 2. Multiple decomposition: a) a circuit partition; b) implementation of multiple decomposition using the method of multiple cutting of BDD

is associated with horizontal cutting of a diagram. Variables that are connected with nodes and that are above the cutting line belong to a bound set. Variables placed below the cutting line create a free set. It turns out that the column multiplicity of the Karnaugh map corresponds to the number of cut nodes, i.e. the nodes placed below the cutting line and whose edges are connected to the nodes from the top part of the diagram. Function decomposition, described by BDD using a single cutting, is illustrated in Fig. 1b [16].

A simple serial decomposition is defined as a decomposition that corresponds to one bound and one free block [2, 7]. In general, the number of inputs of a combinational circuit is high enough and the model is too small. Althoug complex decomposition models have been developed [15, 22], multiple decomposition is considered to be the most interesting approach [7, 13]. This model enables us to create several bound blocks, as shown in Fig. 2a. In the case of a function representation in the form of BDD, there are several methods of searching for a multiple decomposition. The first is based on a multiple cutting, which means that several cutting lines are introduced and the variables placed between given cutting lines are associated with bound sets [12, 13], as illustrated in Fig. 2b. The extracts placed between the lines can be treated as multiroot diagrams (SMTBDD) [13, 14], and can be replaced with a given number of bound functions using the methods presented in [13, 14].

Limitations connected with the number of elements of bound sets are closely associated with the number of inputs of logic blocks in which a function will be mapped [22]. Modern configurable logic blocks include several LUTs that ensures flexibility of a technology mapping process. Usually, the blocks may be configured in many ways. However, on the basis of an analysis of configurabilities of ALM blocks by Intel, configurations may be divided into three groups. The first group involves configurations in which there are two independent LUTs with no common inputs. Appropriate LUTs include a stable number of inputs (appropriate values of *ka* and *kb*) and the implementation of separate (single) functions, as shown in Fig. 3a. The second group consists of configurations in which it is neces-

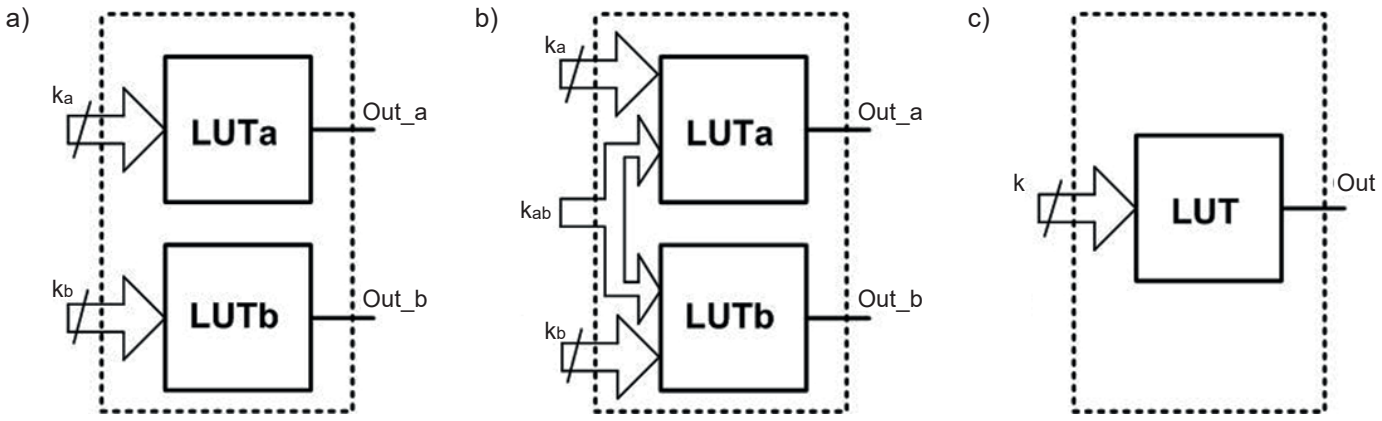

Fig. 3. Groups of configurations of logic blocks in an ALM: a) a group without shared inputs; b) a group with shared inputs; c) a group directed at implementing single functions

sary to share some parts of the inputs between particular LUTs. Figure 3b illustrates the number of common inputs, marked as *kab*. The third group includes configurations in which the whole ALM block implements only one function, usually one that has a higher number of variables (k) than in previous cases. The third case is presented in Fig. 3c.

The flexibility of logic blocks means that it is extremely important to match functions to ALM blocks that are directly connected with the decomposition process of logic functions. Thus, it is key to choose an appropriate configuration for an ALM block that is suitable for a possible partition of a circuit.

3. Disjoint decomposition in the process of technology mapping of combinational circuits

In order to map combinational circuits, we now consider configurations in which there is no sharing of inputs (Fig. 3a, c). We consider configurations in normal mode [10]. It turns out that in the case of ALM blocks in a group shown in Fig. 3a, two configurations can be connected [10]. In the first, $ka = kb = 4$ and in the second, $ka = 5$ and $kb = 3$. In the case of the group of configurations from Fig. 3bc, only one configuration may be found for the ALM blocks for which $k = 6$.

As presented in [13, 14], the technology mapping of combinational circuits in ALM blocks involves fitting bound blocks of a decomposed circuit into LUTs in configurable logic blocks. This mapping is shown in Fig. 4.

The technology mapping shown in Fig. 4a is based on the choice of a cutting line so that the number of variables located above the top cutting line $card(Xb1)$, and the number of variables between the cutting lines $card(Xb2)$ correspond to the number of inputs of the blocks LUTa (ka) and LUTb (kb). The mapping shown in Fig. 4b is based on a choice of the level of cutting so that the number of bound variables $card(Xb)$ is equal to (or at least lower than) the number of inputs of a single block LUT (k) included in a given configuration of an ALM block.

As illustrated in Fig. 4a, multiple decomposition carried out using the multiple cutting method well fits in the group of configurations from Fig. 3a. Decomposition carried out using a single cutting fits in the group of configurations from Fig. 3c. The question arises as to which group and which configuration will be the most effective in this case. This is closely connected to the choice of a decomposition model and the cutting lines of BDD used. A given decomposition model can be described by the number of elements of a bound set $(card(Xb))$ on which the

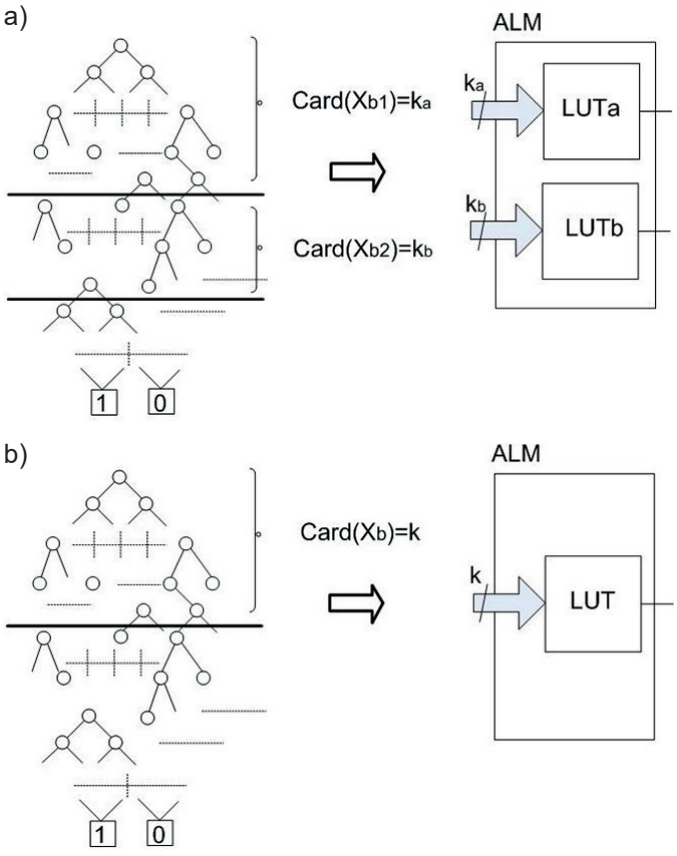

Fig. 4. Technology mapping of bound blocks: a) for multiple decomposition carried out using the multiple cutting method; b) for decomposition carried out using a single cutting of BDD

levels of cutting of BDD depend. It can be also described by the number of bound functions ($numb\_of\_g$) associated with a given decomposition. The various methods for indicating the parameter $numb\_of\_g$ were presented in [15, 24]. Thus, it is necessary to develop algorithms that can enable us to choose a decomposition such that the mappings in one of the configurations in ALM will be the most efficient, taking the number of blocks used into account.

We propose to use triangle tables, as reported in prior literature [14, 22], and to modify these in such a way that the content of certain cells will describe the number of ALM logic blocks needed to implement the subcircuits that are the result of decompositions defined by the pair of parameters: $numb\_of\_g$ and $card(Xb)$. This idea is illustrated in Fig. 5.

A triangle table, as shown in Fig. 5a, corresponds to the configuration *A:* $ka = kb = 4$. A block is configured so that for two independent four-input LUTs, the number of variables in a bound set cannot be higher than four. The triangle table in Fig. 4a is partly filled, because empty cells are left in the columns for which $card(Xb) > 4$. In triangle tables, cases are analysed in which decomposition leads to a limitation on the number of variables. Hence, the maximum number of bound functions introduced is $max(numb\_of\_g) = card(Xb) - 1$. In configuration *A*, a single ALM block may implement two bound functions. The cells associated with the row $numb\_of\_g = 2$ contain the symbol 1*A* (a single ALM block is needed in configuration *A*). The cells in rows $numb\_of\_g = 1$ and $numb\_of\_g = 3$ contain 0.5*A* (only one in two LUTs is used) and 1.5*A* symbols (three LUTs are necessary, i.e. 1.5 of ALM block).

The triangle table in Fig. 5b corresponds to the configuration $ka = 5$ and $kb = 3$. It turns out that for the pair $(card(Xb), numb\_of\_g)$ we can use a LUT that has five inputs (marked in Fig. 5b as *B*), a LUT that has three inputs (*B'*) or both ($B + B'$). A bound set with six bound variables is not used in this configuration, so the appropriate cells in the table are left empty. If a bound set has five or four elements, it is necessary to use an LUT block that has five inputs (configuration *B*). In other cases, for $numb\_of\_g = 1$, any LUT from a block may be used (hence, *B* and *B'* are written in particular cells). However, when $numb\_of\_g = 3$, it is necessary to use both blocks to implement two bound functions (marked as $B + B'$).

The triangle table in Fig. 5c corresponds to the configuration in which the ALM block implements a single function with six variables. In this case, the value in the cells in the table shown in Fig. 5c corresponds to the number of bound functions $numb\_of\_g$. This configuration is marked *C*.

Since each decomposition is accompanied by the pair of numbers $(card(Xb), numb\_of\_g)$, the issue arises of which decomposition should be chosen to match the ALM blocks best. In other words, which ALM configuration (*A, B* or *C*) would be the most effective.

On the basis of the analysis of the tables from Figs 5a, b and c, a results table can be created (a table of technology mapping in ALM) that will be used to choose the most efficient configuration. This results table is presented in Fig. 5d. When analysing these results tables, it can be seen that for the cells associated with $card(Xb) = 6$, the only possibility is to choose configuration *C*. For $card(Xb) = 5$ configurations *C* or *B* may be chosen. It is better to choose configuration *B* since there is a free LUT associated with *B'*. In the case of $card(Xb) = 4$ using configuration *C* is not effective, and thus configurations *A* and *B* may be used. Configuration *A* is a better solution, as it enables us to use unused LUTs with four inputs (in configuration *B*, only LUTs with three inputs are left). Decomposition described using a pair of numbers (3, 4) best corresponds with configuration *A*. In cases where $card(Xb) < 4$, the parameter $numb\_of\_g$ is essential. When its value is 1, it is best to choose configuration *B'*, as an LUT block is available that has five inputs. When $numb\_of\_g = 3$, configurations *A* or $B + B'$ may be chosen. In both cases, a single ALM block is used and there are no free LUTs left.

On the basis of an analysis of the technology mapping table in ALM, a configuration can be chosen that would enable us to reduce as many ALM blocks as possible. For example, if it is possible for a given function to undergo the decomposition described using the following pairs of numbers (6,3), (5,3) or (4,1), we can see that in order to implement this, we need 3,3 and 0.5 ALM blocks, respectively. In this case, the most

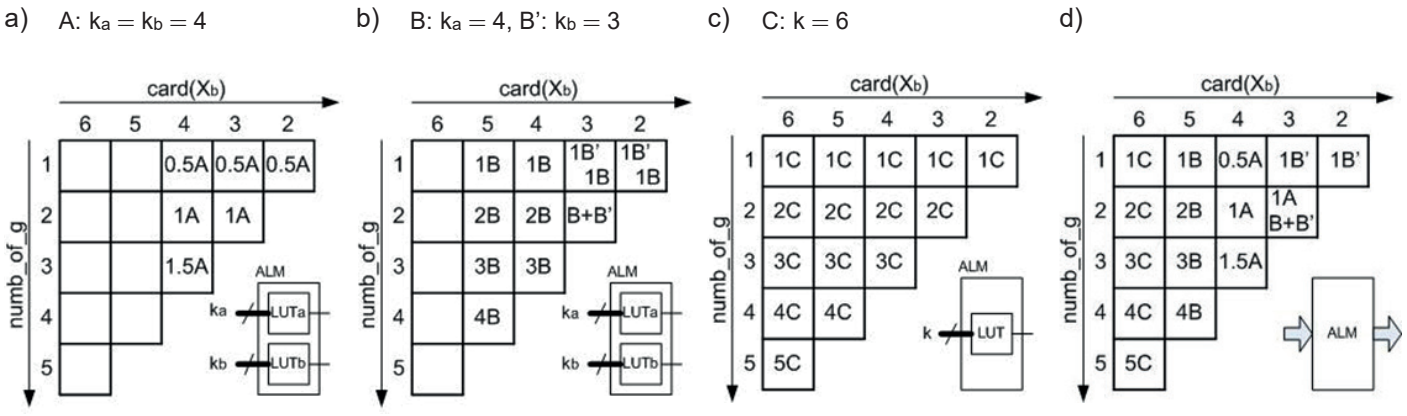

Fig. 5. Triangle tables describing the usage of ALM blocks: a) for configuration A; b) for configuration B; c) for configuration C; d) a technology mapping table in ALM

effective decomposition is that described by the pair of numbers (4,1).

On the basis of an analysis of a technology mapping table in ALM, an algorithm for choosing a configuration can be proposed, as shown in Algorithm 1.

Algorithm 1: Algorithm for choosing a configuration

```
choose_a_configuration (f, set_of_available_ALM_configurations)
{
  for(i = 0; i < set_of_available_ALM_configurations; i++)
    {
      decomposition = indicate_decomposition (cutting_levels (i))
      numb_of_g = decomposition.numb_of_g;
      card(Xb) = decomposition.card(Xb);
       number_of_ALM =
            technology_mapping_table_ALM(numb_of_g, card(Xb))
       if(number_of_ALM < numb_of_ALM_best)
       {
          numb_of_ALM_best = numb_of_ALM;
          configuration_ALM_best = i;
          decomposition_best = decomposition
       }
      }
  return(configuration_ALM_best, decomposition_best)
}
```

4. Mapping of combinational circuits in the process of non-disjoint decomposition

Non-disjoint decomposition is a modification of serial decomposition that was proposed by Asenhurst and Curtis [8, 24]. In a classic model, a bound set and a free set do not have any common elements; in the case of non-disjoint decomposition, some of the variables are attached to both a bound set and a free set. It can be assumed that there is a set of shared variables Xs that includes these variables $Xb \cap Xf = Xs$. It turns out that the variables forming this set can fulfil the role of bound functions, leading to a limitation on the logic blocks needed to implement a bound block. The idea of implementing non-disjoint decomposition is presented in Fig. 6a. Not all variables can belong to the set Xs (i.e. fulfil the role of a switching variable). The idea of searching for switching variables for functions presented in the form of BDD was described in [14, 24].

When analysing configurations of ALM blocks in which sharing of inputs occurs, it can be noticed that they correspond to some basic non-disjoint decompositions, as shown in Fig. 6b. In ALM blocks, there are two such configurations with sharing of inputs: *D:* $ka = 4$, $kb = 3$, $kab = 1$ and *E:* $ka = 3$, $kb = 3$, $kab = 3$. Configuration D can be associated with decomposition in which $card(Xb) = 4$, $card(Xs) = 1$ and $card(Xf) = 3$ or $card(Xb) = 3$, $card(Xs) = 1$ and $card(Xf) = 4$. Configuration E can be associated with decomposition in which $card(Xb) = card(Xf) = 3$ and $card(Xs) = 3$. One of the LUTs in an ALM block is connected with a bound block, and the other is connected with a free block. The necessary condition in both

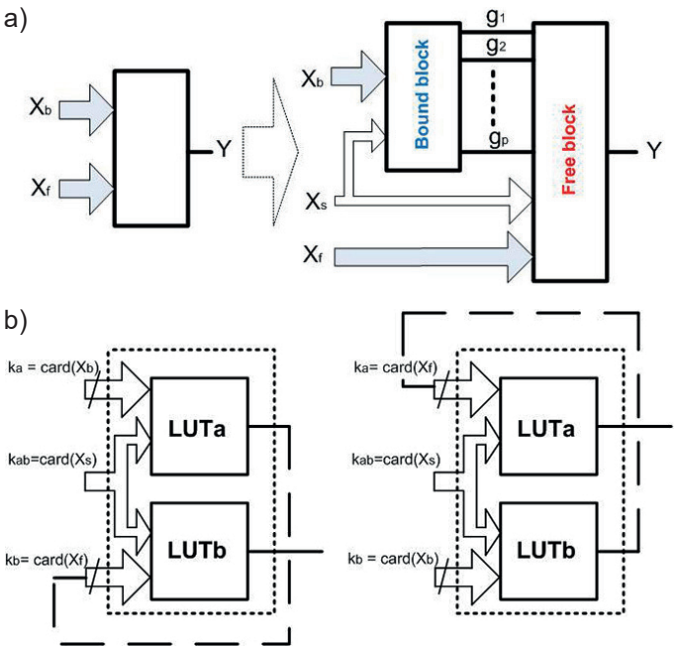

Fig. 6. Non-disjoint decomposition: a) implementation; b) technology mapping in ALM blocks

configurations is the choice of non-disjoint decomposition in which one bound function occurs. The choice of such a decomposition makes it necessary to introduce feedback in an ALM block, which in some cases may be problematic (limitations in next stages of synthesis – placement, routing). Figure 7 presents the example of mapping of non-disjoint decomposition in an ALM block.

Using configurations D and E seems to be reasonable when implementing multi-output functions [15, 22].

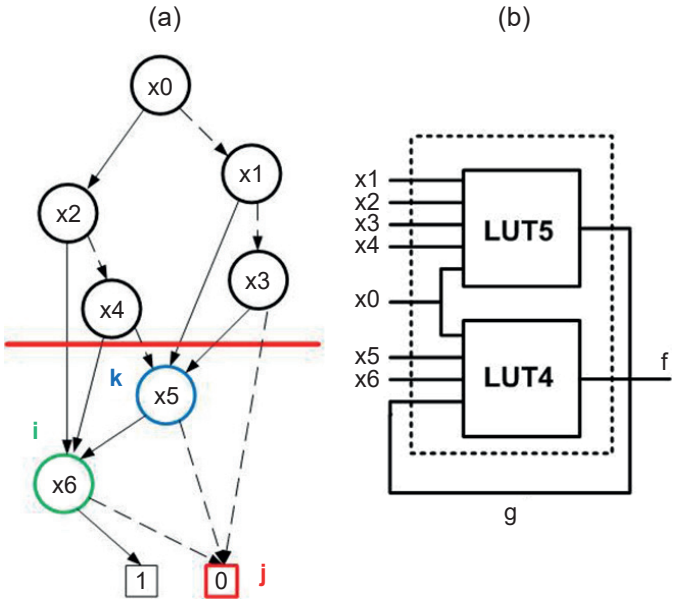

Fig. 7. Example of non-disjoint decomposition: a) BDD; b) technology mapping in ALM blocks

5. Technology mapping of sequential circuits

A sequential circuit is a specific connection of combinational blocks separated with a register, which in the case of FPGA is built from an appropriate number of flip-flops. The specificity of FSM makes the optimisation of technology mapping possible. It is based on the implementation of elements that are at the same time resources of a transition block and an output block. The choice of an appropriate decomposition gives the possibility of an efficient mapping of FSM in ALM blocks.

We consider an exemplary benchmark called beecount [6]. This automaton has three inputs $(n = 3)$, four outputs $(m = 4)$ and seven states $(card(\mathrm{S}) = 7)$. The number of bits needed to code seven states is $k = \lceil lg_2(card(S)) \rceil = 3$. Beecount is Mealy's automaton, for which a general structure is presented in Fig. 8. Apart from a three-bit register $(k = 3)$, two combinational blocks can be distinguished that are described by a transition function $\delta\colon \mathrm{B}^{n+k} \to \mathrm{B}^{k}$ and an output function $\lambda\colon \mathrm{B}^{n+k} \to \mathrm{B}^{m}$. In order to implement a transition block and an output block based on ALM blocks, it is possible to use seven blocks in which the whole ALM block implements only one six-input function. Three ALM blocks can be used to implement a transition function $(\delta\colon \mathrm{B}^{6} \to \mathrm{B}^{3})$, and the rest to implement an output function $(\lambda\colon \mathrm{B}^{6} \to \mathrm{B}^{4})$.

The question arises as to whether there is another way to configure ALM blocks and at the same time to use the resources of FPGA structure efficiently.

The search for an effective mapping starts with the decomposition of a transition function and an output function. It turns out that they belong to many different decompositions. Three decompositions can be distinguished, as presented below.

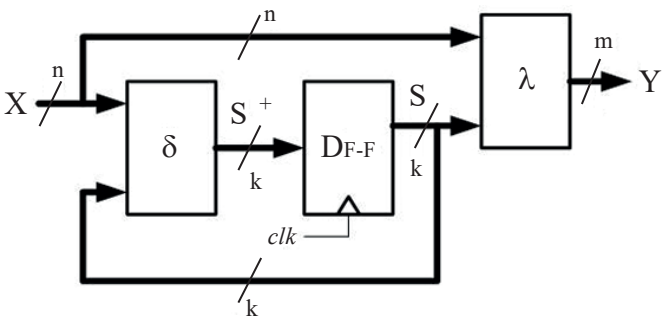

Fig. 8. Block scheme for Mealy FSM

1. $\delta\colon \mathrm{X} \times \mathrm{S} \to \mathrm{S}^{+} \Rightarrow \delta\colon \{\mathrm{i2}, \mathrm{i1}, \mathrm{i0}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}\} \to \{\mathrm{q2}, \mathrm{q1}, \mathrm{q0}\}$

$\delta_1\colon \{\mathrm{i2}, \mathrm{i1}, \mathrm{i0}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}\} \to \{\mathrm{q2}\}$

$\mathrm{X_b} = \{\mathrm{i2}, \mathrm{i1}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}\}; \quad \mathrm{X_f} = \{\mathrm{i0}\};$

$\nu(\mathrm{i0} \,|\, \mathrm{i2}, \mathrm{i1}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}) = 2 \quad \Rightarrow numb\_of\_g = 1$

$\delta_2\colon \{\mathrm{i2}, \mathrm{i1}, \mathrm{i0}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}\} \to \{\mathrm{q1}, \mathrm{q0}\}$

$\mathrm{X_b} = \{\mathrm{i1}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}\}; \quad \mathrm{X_f} = \{\mathrm{i2}, \mathrm{i0}\};$

$\nu(\mathrm{i2}, \mathrm{i0} \,|\, \mathrm{i1}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}) = 4 \quad \Rightarrow numb\_of\_g = 2$

2. $\lambda\colon \mathrm{X} \times \mathrm{S} \to \mathrm{Y} \Rightarrow \lambda\colon \{\mathrm{i2}, \mathrm{i1}, \mathrm{i0}, \mathrm{Q2}, \mathrm{Q1}, \mathrm{Q0}\} \to \{\mathrm{o3}, \mathrm{o2}, \mathrm{o1}, \mathrm{o0}\}$

$\mathrm{X_b} = \{\mathrm{i2}, \mathrm{i1}, \mathrm{i0}, \mathrm{Q2}, \mathrm{Q}\}; \quad \mathrm{X_f} = \{\mathrm{Q0}\};$

$\nu(\mathrm{Q0} \,|\, \mathrm{i2}, \mathrm{i1}, \mathrm{i0}, \mathrm{Q2}, \mathrm{Q1}) = 8 \quad \Rightarrow numb\_of\_g = 3$

Finding and choosing the decompositions above allows for an effective mapping of FSM in ALM blocks, as illustrated in Fig. 9.

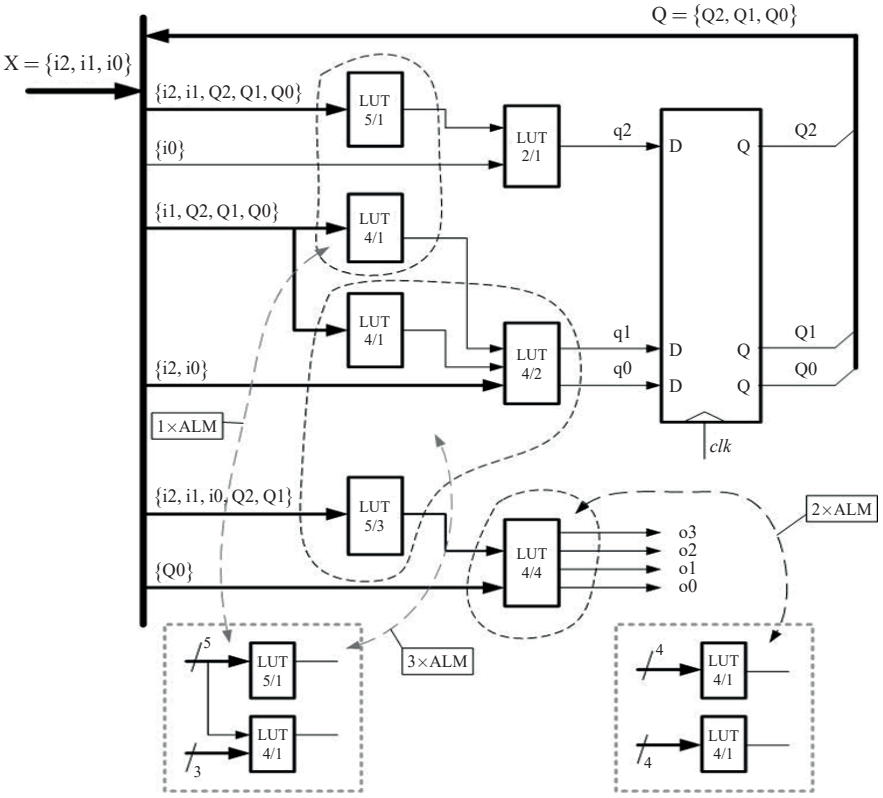

Fig. 9. Technology mapping of the beecount automaton in ALM blocks

The technology mapping presented in Fig. 9 uses four ALM blocks that work in a configuration with one shared input (LUT 5/1 + LUT 4/1 – one common input) and two ALM blocks in a configuration without sharing of inputs (LUT 4/1 + LUT 4/1). It is necessary to use an LUT 2/1 block that can be implemented in each configuration, leaving resources which can be used in other ways.

6. Experimental results

A prototypical software module called MultiDec (MD) was created for the purpose of experiment, and was used to decompose logic functions. A series of experiments were conducted to prove effectiveness of the ideas presented in this paper. The synthesis was carried out using a commercial tool called Quartus II, produced by Intel. A set of benchmarks was synthesised [6], and these were described in the form of equations in Verilog HDL. Descriptions were generated from .pla (withoutMD) or from the descriptions .pla, which were initially decomposed based on the methods of technology mapping in the tool in [13, 14] (withMD). The main goal of this research was to compare the results of synthesis obtained for two series of experiments.

In the first series of experiments, combinational circuits were analysed, and the results are shown in Table 1. Table 1 includes the number of ALM blocks and the number of particular LUT (ALUT) blocks that have a given number of inputs included in the ALM blocks. In addition, the number of logic levels (depth) is given. The last row shows the sum of the number of levels and the number of separate blocks. Moreover, the total number of blocks is presented in the form of a graph in Fig. 10.

Table 1
The results of synthesis for combinational circuits

			without MD							with MD						
	In	Out	ALM	ALUT7	ALUT6	ALUT5	ALUT4	ALUT <= 3	depth	ALM	ALUT7	ALUT6	ALUT5	ALUT4	ALUT <= 3	depth
5xp1	7	10	11	0	5	3	2	6	2	9	0	4	1	1	7	2
alu2	10	8	15	0	5	11	5	4	2	13	1	4	6	3	6	3
b12	15	9	10	1	4	3	3	4	2	12	0	5	8	1	5	3
cm163a	16	5	5	1	2	2	1	0	2	7	1	3	3	1	2	3
f51m	8	8	12	0	5	4	5	5	3	6	0	2	3	2	3	2
Inc	7	9	11	0	8	2	1	2	2	11	0	8	2	1	3	3
Ldd	9	19	17	0	4	4	10	11	4	19	0	8	2	7	13	2
misex1	8	7	6	2	3	0	2	0	1	7	0	4	2	4	0	2
misex2	25	18	18	1	6	9	8	4	3	20	0	12	8	3	5	3
Pcle	19	9	9	1	5	2	1	2	2	9	2	4	2	1	2	3
rd73	7	3	8	0	6	0	0	3	2	3	0	0	3	2	1	2
rd84	8	4	52	2	32	11	12	12	4	7	0	4	1	1	3	3
Sct	19	15	12	0	3	5	7	5	2	14	1	3	9	5	6	3
Sqn	7	3	8	0	6	0	0	3	2	8	0	6	0	0	3	2
sqr6	6	11	7	0	4	3	1	2	1	8	0	4	3	1	3	2
sqrt8	8	4	2	0	2	0	1	4	3	2	0	4	1	1	2	2
t481	16	1	3	0	0	0	5	1	3	6	0	2	5	2	0	4
x2	10	7	8	0	2	3	2	6	2	10	0	3	6	4	4	3
Sum:			**214**	**8**	**102**	**62**	**66**	**74**	**42**	**171**	**5**	**80**	**65**	**40**	**68**	**47**

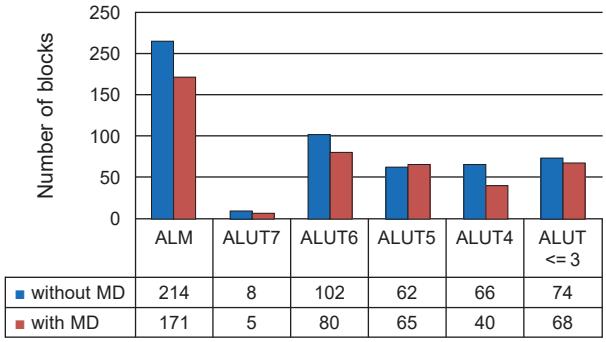

Fig. 10. Number of logic blocks obtained after the synthesis of combinational circuits

From an analysis of the graph in Fig. 10, it can be seen that the proposed methods of technology mapping led to a substantial reduction in the number of ALM blocks. The number of the various integral LUT blocks was reduced (apart from LUT5), and the greatest reduction was obtained for LUTs with six and four inputs. It should be emphasised that as a result of using the proposed methods, the number of logic levels was increase by about 10%, which is unfavourable from the point of view of the dynamic behavior of a circuit.

In the second series of experiments, combinational blocks of sequential circuits were synthesised, and the blocks d and l were synthesised separately. It was assumed coding of inner states and using a natural binary code. The results obtained for the d blocks are shown in Table 2, and the those for the l blocks in Table 3.

Table 2
Results of the synthesis of transition blocks (δ function)

			delta													
			without MD							with MD						
	In	Out	ALM	ALUT7	ALUT6	ALUT5	ALUT4	ALUT <= 3	depth	ALM	ALUT7	ALUT6	ALUT5	ALUT4	ALUT <= 3	depth
Beecount	6	3	3	0	3	0	0	0	1	3	0	3	0	0	0	1
dk14	6	3	3	0	3	0	0	0	1	2	0	0	0	2	1	1
dk15	5	2	1	0	0	2	0	0	1	1	0	0	0	2	0	1
dk16	7	5	13	0	10	0	0	5	2	12	0	9	1	0	5	2
dk17	10	8	11	0	5	3	6	3	2	7	0	3	2	5	1	2
dk27	4	3	2	0	0	0	3	0	1	2	0	0	0	2	1	1
dk512	5	4	2	0	0	4	0	0	1	2	0	0	4	0	0	1
Donfile	7	5	2	0	4	0	2	2	2	6	0	4	0	2	1	2
ex2	7	5	12	0	9	0	0	5	2	11	0	6	4	1	4	3
ex3	6	4	4	0	4	0	0	0	1	4	0	4	0	0	0	1
ex4	10	4	7	3	2	0	2	1	2	9	0	5	5	1	2	3
ex5	6	4	4	0	4	0	0	0	1	4	0	4	0	0	0	1
ex7	6	4	4	0	4	0	0	0	1	4	0	3	1	0	0	1
Lion	4	2	1	0	0	0	2	0	1	1	0	0	0	1	1	1
lion9	6	4	4	0	4	0	0	0	1	4	0	4	0	0	0	1
Mc	5	2	1	0	0	1	0	1	1	1	0	0	1	0	1	1
s8	7	3	7	0	4	2	1	2	2	5	0	2	5	0	1	2
s27	7	3	3	0	2	1	0	0	1	3	0	2	1	0	1	2
Shiftreg	4	3	2	0	0	0	2	1	1	2	0	0	0	2	1	1
Tav	6	2	1	0	0	0	0	2	1	1	0	0	0	0	2	1
train4	4	2	1	0	0	0	1	1	1	1	0	0	0	1	1	1
train11	6	4	4	0	4	0	0	0	1	4	0	4	0	0	0	1
Sum:			**92**	**3**	**62**	**13**	**19**	**23**	**28**	**89**	**0**	**53**	**24**	**19**	**23**	**31**

Table 3
Results of the synthesis of output blocks (l function)

			lambda													
			without MD							with MD						
	In	Out	ALM	ALUT7	ALUT6	ALUT5	ALUT4	ALUT <= 3	depth	ALM	ALUT7	ALUT6	ALUT5	ALUT4	ALUT <= 3	depth
beecount	6	4	4	0	4	0	0	0	1	4	0	4	0	0	0	1
dk14	6	5	5	0	5	0	0	0	1	4	0	3	0	1	1	1
dk15	5	5	3	0	0	5	0	0	1	3	0	0	5	0	0	1
dk16	7	3	5	0	2	2	1	2	2	3	1	0	3	1	0	2
dk17	10	3	7	0	1	6	4	1	2	4	0	3	1	0	0	2
dk27	4	2	1	0	0	0	1	1	1	1	0	0	0	1	1	1
dk512	5	3	2	0	0	2	0	1	1	2	0	0	2	0	1	1
Donfile	7	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1
ex2	7	2	3	0	2	0	0	1	2	2	0	1	0	2	0	2
ex3	6	2	2	0	2	0	0	0	1	2	0	2	0	0	0	1
ex4	10	9	7	0	5	4	0	0	1	7	0	5	4	0	0	1
ex5	6	2	1	0	1	0	0	0	1	1	0	1	0	0	0	1
ex7	6	2	1	0	1	0	0	0	1	1	0	1	0	0	0	1
Lion	4	1	1	0	0	0	1	0	1	1	0	0	0	1	0	1
lion9	6	1	1	0	1	0	0	0	1	1	0	1	0	0	0	1
Mc	5	5	2	0	0	1	0	3	1	2	0	0	1	0	3	1
s8	7	1	1	0	0	0	2	0	2	1	0	0	0	2	0	2
s27	7	1	2	0	1	0	0	1	2	2	0	1	0	0	1	2
Shiftreg	4	1	1	0	0	0	0	1	1	1	0	0	0	0	1	1
Tav	6	4	4	0	4	0	0	0	1	2	0	0	4	0	0	1
train4	4	1	1	0	0	0	1	0	1	1	0	0	0	1	0	1
train11	6	1	1	0	1	0	0	0	1	1	0	1	0	0	0	1
Sum:			**56**	**0**	**30**	**20**	**10**	**12**	**27**	**47**	**1**	**23**	**20**	**9**	**9**	**27**

The results of the synthesis of sequential circuits are presented in a synthetic form in the graphs in Figs 11 and 12.

From the point of view of the experiments, the key problem in the synthesis of combinational blocks of FSMs is that the circuits are so small that they do not need decomposition. In the case of blocks with a higher number of inputs, using decomposition together with the proposed methods of technology mapping gives (for blocks δ and λ) a reduction in the number of ALM blocks needed. In both cases, the biggest reduction is obtained for LUTs with six inputs. An analysis of the obtained

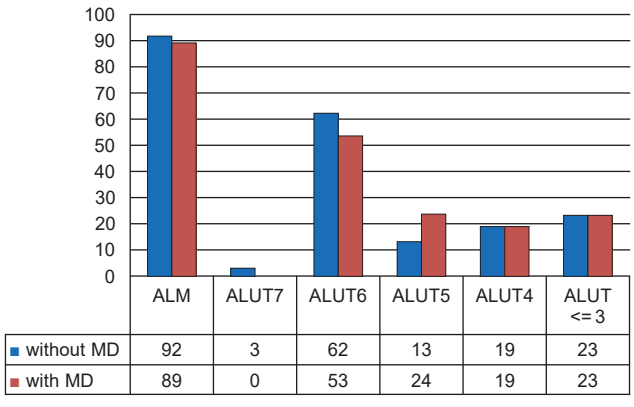

Fig. 11. Number of logic blocks obtained after the synthesis of the δ blocks

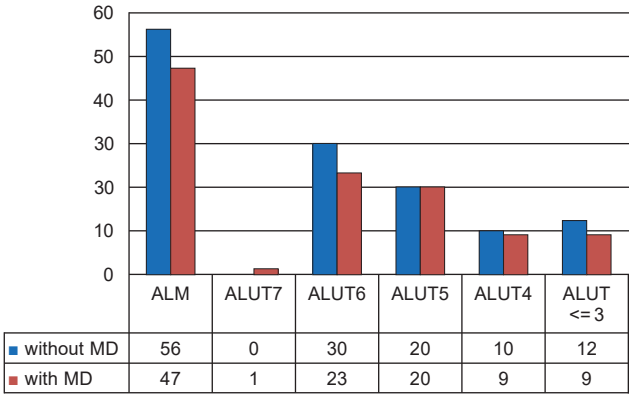

Fig. 12. Number of logic blocks obtained after the synthesis of the λ blocks

number of logic levels shows that in the case of the δ blocks, there is a rapid growth in the number of levels when the proposed methods are used. For the λ block, the number of levels remains the same.

The most important indicator confirming the advantages of the proposed method is the fact that the total number of ALM blocks is lower when using the proposed method of technology mapping. The situation is the same for both combinational and sequential circuits. Thus, a key element in the technology mapping of the circuits implemented in FPGA is the ability to use available configurations of ALMs.

In addition, MultiDec was compared with a leading academic system called ABC [3] (&get; &st; &synch2; &if-K6; &ps;). The results are presented in Tables 4–6 for the set of

Table 4
Comparison of MultiDec with ABC for combinational circuits

			MultiDec				ABC			
	In	Out	Blocks	Quartus	depth	Time	Blocks	Quartus	depth	Time
5xp1	7	10	8	9	2	296	13	9	2	220
alu2	10	8	15	13	4	764	17	10	3	190
b12	15	9	12	12	2	655	15	10	2	120
cm163a	16	5	6	7	2	561	7	6	2	160
f51m	8	8	6	6	2	124	17	10	3	160
inc	7	9	9	11	2	93	13	12	2	140
ldd	9	19	19	19	2	1560	25	14	2	140
misex1	8	7	7	7	2	140	9	6	2	140
misex2	25	18	23	20	3	3166	32	18	2	140
pcle	19	9	9	9	3	3915	12	9	2	120
rd73	7	3	3	3	2	140	12	6	3	190
rd84	8	4	6	7	2	312	37	24	3	190
sct	19	15	14	14	3	1747	17	11	2	120
sqn	7	3	5	8	2	124	11	8	2	140
sqr6	6	11	7	8	1	31	10	7	1	160
sqrt8	8	4	5	2	2	78	9	6	3	140
t481	16	1	5	6	4	608	20	3	3	140
x2	10	7	6	10	2	249	13	8	2	130
Sum:			**165**	**171**	**42**	**14563**	**289**	**177**	**41**	**2740**

Table 5
Comparison of MultiDec with ABC (δ function)

			delta							
			MultiDec				ABC			
	In	Out	Blocks	Quartus	depth	Time	Blocks	Quartus	depth	Time
Beecount	6	3	3	3	1	15	3	3	1	260
dk14	6	3	3	2	1	15	3	3	1	230
dk15	5	2	1	1	1	15	2	1	1	160
dk16	7	5	12	12	2	265	18	13	2	200
dk17	10	8	12	7	2	1466	19	11	2	160
dk27	4	3	2	2	1	15	3	2	1	170
dk512	5	4	2	2	1	15	4	2	1	140
Donfile	7	5	6	6	2	62	7	6	2	160
ex2	7	5	8	11	2	202	14	12	2	190
ex3	6	4	4	4	1	15	4	4	1	140
ex4	10	4	6	9	2	312	9	7	2	140
ex5	6	4	4	4	1	31	4	4	1	140
ex7	6	4	4	4	1	31	4	4	1	140
Lion	4	2	1	1	1	15	2	1	1	120
lion9	6	4	4	4	1	15	4	4	1	140
Mc	5	2	1	1	1	0	2	1	1	130
s8	7	3	4	5	2	140	8	7	2	130
s27	7	3	3	3	1	15	3	3	1	140
Shiftreg	4	3	2	2	1	15	3	2	1	110
Tav	6	2	1	1	1	15	1	1	1	140
train4	4	2	1	1	1	15	2	1	1	120
train11	6	4	4	4	1	31	4	4	1	130
Sum:			**88**	**89**	**28**	**2720**	**123**	**96**	**28**	**3390**

Table 6
Comparison of MultiDec with ABC (λ function)

			lambda							
			MultiDec				ABC			
	In	Out	Blocks	Quartus	depth	Time	Blocks	Quartus	depth	Time
Beecount	6	4	4	4	1	46	4	3	1	220
dk14	6	5	5	4	1	46	5	5	1	170
dk15	5	5	3	3	1	31	5	3	1	110
dk16	7	3	5	3	2	78	6	5	2	140
dk17	10	3	5	4	2	280	10	7	3	130
dk27	4	2	1	1	1	15	2	1	1	160
dk512	5	3	2	2	1	15	3	2	1	190
Donfile	7	1	1	1	1	15	1	1	1	80
ex2	7	2	2	2	2	78	3	3	2	140
ex3	6	2	2	2	1	15	2	2	1	130
ex4	10	9	7	7	1	21	9	8	1	140
ex5	6	2	2	1	1	31	1	1	1	140
ex7	6	2	2	1	1	15	1	1	1	130
lion	4	1	1	1	1	15	1	1	1	110
lion9	6	1	1	1	1	15	1	1	1	140
Mc	5	5	1	2	1	15	3	2	1	120
s8	7	1	1	1	2	46	2	1	2	140
s27	7	1	1	2	1	15	1	1	1	140
Shiftreg	4	1	1	1	1	15	1	1	1	120
Tav	6	4	4	2	1	15	4	2	1	120
train4	4	1	1	1	1	15	1	1	1	120
train11	6	1	1	1	1	15	1	1	1	120
Sum:			**53**	**47**	**26**	**852**	**67**	**53**	**27**	**3010**

benchmarks presented above. In the appropriate columns of these tables, the following parameters were determined: the number of logic blocks predicted by decomposition tools ('Blocks'); the number of ALM blocks obtained after a synthesis in Quartus from .pla descriptions generated by appropriate decomposition tools ('Quartus'); the number of logic levels ('depth'); and the synthesis times using the academic tools, expressed in [ms].

When comparing both systems, taking into consideration the number of logic blocks needed to implement separate benchmarks, it is noticeable that the number of blocks after decomposition ('Blocks') is lower than that obtained after last synthesis stages in Quartus ('Quartus'). In this case, the results obtained for MultiDec are slightly better, but are substantially worse than the predictions obtained in MultiDec. The results of synthesis indicate the substantial advantages of using the configurabilities of blocks that effectively use MultiDec. The obtained results are almost the same when the number of logic levels is taken into account. When decomposition times are compared, it can be seen that ABC is much better (as shown in Table 4).

7. Conclusion

Contemporary programmable circuits include very flexible blocks, whose logic resources are not always used properly. The cause of this problem includes the implementation of inefficient methods of technology mapping in synthesis tools. Synthesis algorithms should take into consideration the possibility of configuring logic blocks, as this is the only effective way of implementing circuits.

This paper presents the idea of technology mapping of digital circuits including configurable abilities of logic blocks. The essence of this idea is shown using the example of ALM blocks, but this is a general idea that can be used for other families of programmable circuits. Our experimental results demonstrate its effectiveness for two kinds of circuits: combinational and sequential.

The major disadvantage of the solution presented here is its limited scalability, although the method works well in the case of small circuits that have a low number of inputs. Thus, the proposed methods can be used locally for separate parts of a larger circuit (decomposed using other methods e.g. a parti-

tion of an AIG graph). This may result in the reduction of the number of logic blocks.

An original way of choosing a decomposition path is to use triangle tables that are connected with particular configurations of ALM blocks. This enables us to use the logic resources of a LUT-based FPGA efficiently. Unfortunately, the main drawback of minimising the number of ALM blocks used, i.e. the area of a circuit, is the expansion in the number of logic levels. Thus, it is necessary to search for decomposition strategies that could enable us to reduce the number of logic levels. This problem is the topic of the present research whose aim is to develop methods of technology mapping leading to reduction of the area and at the same time caring for dynamic features of obtained solutions.

Acknowledgements. The study was supported partially by the Polish Ministry of Science and Higher Education.

References

[1] S.B. Akers, "Binary decision diagrams", *IEEE Transactions on Computers*, C-27(6), 509–516 (1978).
[2] R.L. Ashenhurst, "The decomposition of switching functions", *Proceedings of the International Symposium on the Theory of Switching*, 1957.
[3] Berkeley Logic Synthesis Group: *ABC: A System for Sequential Synthesis And VerificAtion*, Dec. 2005 [Online]. Available: http://www.eecs.berkeley.edu/~alanmi/abc
[4] J.A. Brzozowski and T. Łuba: *Decomposition of Boolean Functions Specified by Cubes*, Journal of Multi-Valued Logic & Soft Computing, vol. 9, pp. 377–417, Old City Publishing Inc., Philadelphia 2003.
[5] D. Chen and J. Cong, "DAOmap: A depth-optimal area optimization mapping algorithm", in *Proc. ICCAD*, pp. 752–759, (2004).
[6] Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University, http://www.cbl.ncsu/edu/.
[7] H.A. Curtis, *The Design of Switching Circuits*, D. van Nostrand Company, Inc., Princeton, New Jersey, Toronto, New York, 1962.
[8] E. Dubrova, "A polynominal time algorithm for non-disjoint decomposition of multi-valued functions", 34*th International Symposium on Multiple-Valued Logic*, 309–314 (2004).
[9] I. Háleček, P. Fišer, and J. Schmidt, "Towards AND/XOR balanced synthesis: Logic circuits rewriting with XOR", *Microelectronics ReliAbility*, 81, 274–286 (2018).
[10] Intel Stratix 10 Logic Array Blocks and Adaptive Logic Modules User Guide, UG-S10LAB, 2017.
[11] S. Jang, B. Chan, K. Chung, and A. Mishchenko, "WireMap: FPGA technology mapping for improved routability and enhanced LUT merging", *ACM Trans. Reconfigurable Technology and Systems (TRETS)*, 2(2), Article 14 (2009).
[12] M. Kubica and D. Kania, "SMTBDD : new form of BDD for logic synthesis", *International Journal of Electronics and Telecommunications*, 62(1), 33–41 (2016).
[13] M. Kubica and D. Kania, "Area-oriented technology mapping for LUT-based logic blocks", *International Journal of Applied Mathematics and Computer Science*, 27 (1), 207–222 (2017).
[14] M. Kubica and D. Kania, "Decomposition of multi-output functions oriented to configurability of logic blocks", *Bull. of the Pol. Ac.: Tech.* 65(3) 317–331 (2017).
[15] M. Kubica, A. Opara, and D. Kania, "Logic synthesis for FPGAs based on cutting of BDD, Microprocessor and Microsystems", 52, 173–187 (2017).
[16] Y-T. Lai, M. Pedram, and S. Vrudhula, "BDD based decomposition of logic for functions with applications to FPGA synthesis", in *Proc. of Design Automation Conf*., pp. 642–647, 1993.
[17] T. Łuba, G. Borowik, and A. Kraśniewski, "Synthesis of finite state machines for implementation with programmable structures", *Electronics and Telecommunications Quarterly*, 55/2009 (2), 183–200 (2009).
[18] S. Minato, *Binary Decision Diagrams and Applications for VLSI CAD*, Kluwer Academic Publishers, 1996.
[19] A. Mishchenko, R. Brayton, W. Feng, and J. Greene, "Technology mapping into general programmable cells", *Proc. FPGA*'15.
[21] A. Mishchenko, S. Cho, S. Chatterjee, and R. Brayton, "Combinational and sequential mapping with priority cuts", in *Proceedings of the* 3007 *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '07),* 354–361, 2007.
[22] A. Mishchenko, S. Chatterjee, and R. Brayton, "Improvements to technology mapping for LUT-based FPGAs", IEEE TCAD, 26(2), 240–253, 2007.
[23] A. Opara, M. Kubica, and D. Kania, "Strategy of logic synthesis using MTBDD dedicated to FPGA", *Integration: The VLSI Journal* 62, 142–158 (2018).
[24] S. Ray et al., "Mapping into LUT structures" in *Design, Automation and Test in Europe*, Dresden, Germany, pp. 1579–1584, 2012.
[25] C. Scholl, *Functional Decomposition with Application to FPGA Synthesis*, Kluwer Academic Publisher, Boston, 2001.
[26] P. Szotkowski, M. Rawski, and H. Selvaraj, "A graph-based approach to symbolic functional decomposition of finite state machines", *Systems Science*, 35 (2), 41–47, 2009.
[27] 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).
[28] 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).