sciendo

Int. J. Appl. Math. Comput. Sci., 2023, Vol. 33, No. 2, 267–284
DOI: 10.34768/amcs-2023-0020

amcs

TECHNOLOGY MAPPING OF MULTI–OUTPUT FUNCTIONS LEADING TO THE REDUCTION OF DYNAMIC POWER CONSUMPTION IN FPGAs

ADAM OPARA [a,*], MARCIN KUBICA [b]

[a]Department of Graphics, Computer Vision and Digital Systems
Silesian University of Technology
ul. Akademicka 2A, 44-100 Gliwice, Poland
e-mail: Adam.Opara@polsl.pl

[b]Department of Digital Systems
Silesian University of Technology
ul. Akademicka 2A, 44-100 Gliwice, Poland
e-mail: Marcin.Kubica@polsl.pl

This article presents a synthesis strategy aimed at minimizing the dynamic power consumption of combinational circuits mapped in LUT blocks of FPGAs. The implemented circuits represent the mapping of multi-output functions. Properly selected multi-output functions are described using a new form of the binary decision diagram (BDD), which is an extension of pseudomulti-terminal BDDs (PMTBDDs) in the literature. The essence of limiting power consumption is to include additional parameters during decomposition, such as the switching activity associated with the switching PMTBDD (SW-PMTBDD). In addition, we highlight the key importance of circuit optimization methods via non-disjoint decomposition when minimizing power consumption. An algorithm is proposed to assess the effectiveness of decomposition, considering several parameters, such as the number of non-disjoint decompositions as well as that of shared and non-shared bound functions or the switching activity. The results of experiments that demonstrate the effectiveness of the proposed methods are also included.

Keywords: low power synthesis, FPGA, switching activity, decomposition, technology mapping.

1. Introduction

Reducing power consumption in digital devices is becoming increasingly important, and well-designed and optimized devices effectively reduce the requirements of their power sources. This process has opened up new spaces for complex digital devices, such as the Internet of things (IoT) or, more broadly, cyber-physical systems (CPSs) (Wojnakowski *et al*., 2021; Wisniewski, 2021; Patalas-Maliszewska *et al*., 2022; Wisniewski *et al*., 2020). Naturally, the reduction of power consumption improves user comfort with mobile devices (e.g., longer battery life). These issues make us consider how digital devices should be designed to be efficient in terms of power consumption.

There are many ways to reduce power consumption. The simplest approach is to temporarily disable modules that are not being currently used. Another solution is the employment of specific functions in a hardware-software manner (Benini and Micheli, 2000), which can be implemented as a high-level synthesis (Raghunathan *et al*., 2012; Ali and Al-Hashimi, 2007; Bard and Rafla, 2008; Brooks *et al*., 2000; Kim and Kim, 2000). Additionally, techniques exist to reduce power consumption at a low level of synthesis. These methods include local voltage reduction (Chen *et al*., 2001; Manzak and Chakrabarti, 2002), "power gating" (Kim and Kim, 2000) and techniques related to limiting the clock frequency, which negatively affects the execution time of the implemented functions. Therefore, a good alternative is "clock gating" (Kuc *et al*., 2020) or implementing the circuit in the form of a GALS (globally asynchronous locally synchronous) structure. The essence of GALS is the implementation of the global asynchronous structure, in which synchronous structures with an appropriately

*Corresponding author

adjusted clock frequency are locally located.

Many complex digital circuits are implemented in FPGA (field programmable gate array) devices. This approach allows easy prototyping of circuits and ensures good time efficiency of the implementation of specific functions. The growing popularity of FPGAs leads to the development of dedicated synthesis strategies targeted at these devices. The goal of these strategies is to obtain effective solutions in terms of limiting the use of logic resources (Opara *et al.*, 2018; 2019; Rawski *et al.*, 2005; Selvaraj *et al.*, 2006; Vermuri *et al.*, 2002; Jóźwiak and Chojnacki, 2003; Ling *et al.*, 2005; Li *et al.*, 2022) or speed of operation (Cheng *et al.*, 2008).

The problem of mapping logic circuits with the use of 3-input NPN (negation/permutation/negation) classes has been presented by Marakkalage *et al*. (2020). There are also strategies to reduce power consumption (Chung and Brayton, 2009; Kubica *et al*., 2021a; Lin *et al*., 2022). The key is therefore to identify methods reducing power consumption that can be used in FPGA-oriented synthesis. In the case of combinational circuits, the methods of reducing power consumption are shown by BLSG (2005), Sánchez *et al*. (2009), Lindgren *et al*. (2001), Balasubramanian and Anantha (2007), or Mehrotra (2013). However, in the case of sequential circuits, the problem of power consumption was discussed by Barkalov *et al.* (2020b; 2020b; 2021; 2022). Additionally, Kajstura and Kania (2018) as well as Kubica *et al*. (2018) proposed a method associated with the appropriate coding of FSM internal states, which leads to a reduction in the number of circuit switches and thus a reduction in dynamic power consumption.

The goal of this article is to present a new strategy that aims to reduce dynamic power consumption of structures implemented in FPGAs. This strategy is intended for effective implementation of the multi-output functions, which will ensure the sharing of logical resources. The key element of this strategy is the decomposition of multi-output functions, considering the necessity to reduce the switching activity of the implemented circuits.

This article is an extension of the methods developed by the authors that focus on power. The authors have adapted these methods to implement a power minimization strategy, and some of the ideas featured are an extension of those initially presented by Kubica *et al*. (2021a). The primary contributions of the article are (i) adapting various forms of BDDs describing sets of functions to the power minimization process, and (ii) proposing a power minimization process strategy using the description of the set of functions in the form of a BDD. Additionally, the authors describe an improved algorithm for dividing functions into clusters.

The paper includes the following: a theoretical introduction showing the essence of decomposition with the use of the BDD; a section presenting the problem of the initial division of functions into clusters and a section presenting the essence of writing a multi-output function in the form of a PMTBDD; a new diagram type, SWPMTBDD, that accounts for the switching activity for the description of a multi-output function; a decomposition method that aims to reduce dynamic power consumption, which is the essence of the synthesis strategy presented in the next section; a presentation of the results of the experiments performed; and conclusions drawn from the results of this study.

2. Theoretical background

The key element of FPGA-oriented synthesis is the decomposition of functions related to the technology mapping of functions in the logic resources of these devices. These resources are commonly LUT (look-up table) blocks, which can implement any logic function with a limited (small) number of k variables. Function decomposition is a mathematical model of the division of the implemented structure between these blocks. Naturally, the method of representing the logic function is important for the implementation of decomposition. Efficient, in terms of memory use and speed of operation implementation, is the representation of functions in the form of a BDD (binary decision diagram) (Akers, 1978).

Function decomposition theory dates back to the mid-twentieth century, as shown by Ashenhurst (1957) and further developed by Curtis (1962). The simplest decomposition model is simple serial decomposition. In this model, all variables of the function are divided into two disjoint sets: the bound set X_b and the free set X_f. In the logic structure, individual sets are implemented in separate blocks: a bound block and a free block (Fig. 1). These blocks are connected to each other by the $numb\_of\_g$ lines. These connections are associated with the bound functions g. The bound functions are generated in a bound block based on the variables in the bound set X_b. Conversely, the target free function f is generated in a free block based on variables from the set X_f and the values of individual bound functions. From the perspective of the efficiency of the obtained solutions, the goal of this method is to limit the number of functions g. Therefore, it becomes critical to efficiently determine the number of these functions ($numb\_of\_g$) when selecting variables for individual sets.

With the representation of a function in the form of a BDD (the authors mean the reduced and ordered form—an ROBDD), the decomposition is performed by cutting the diagram horizontally (Minato, 1996; Scholl, 2001). The upper part of the diagram, which is above the cut line, corresponds to the bound set X_b. The lower part of the diagram (below the cut line) corresponds to the free set X_f. With a BDD, to determine the number

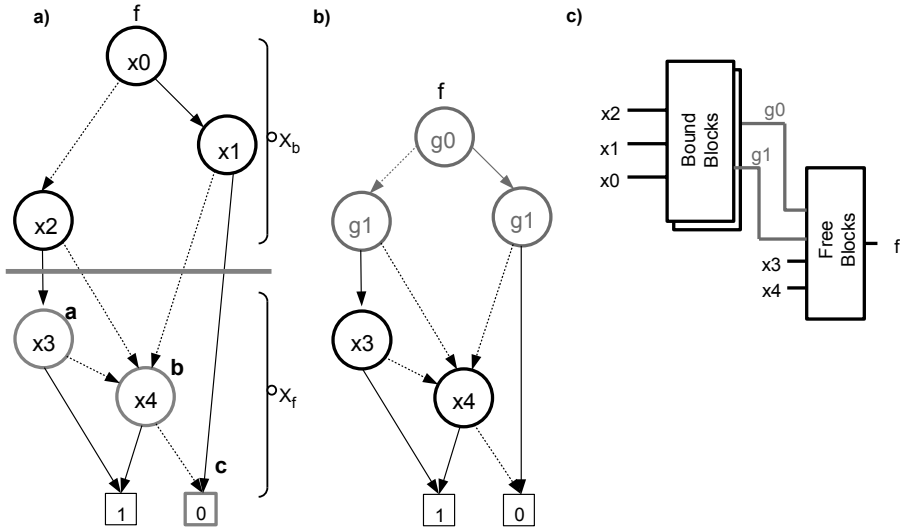


Fig. 1. Simple serial decomposition: BDD cut (a), obtained BDD after decomposition (b), partition model for simple serial decomposition (c).

of necessary linking functions g, it is necessary to know the number of the so-called cut nodes (i.e., the number of g must be large enough to distinguish between the individual cut nodes). The cut nodes are the nodes in the lower part of the diagram that have the edges from the upper part brought together. The idea of decomposition with a BDD is shown in Fig. 1.

Considering the example from Fig. 1(a), three cut nodes can be distinguished in the lower part of the diagram (marked in grey) as a result of the cut. To distinguish them, it is necessary to introduce two bound functions: g_0 and g_1. Therefore, in Fig. 1(b), the top part of the diagram has been replaced with nodes associated with the bound functions (marked in grey), which leads to the partition shown in Fig. 1(c). The partition model associated with simple serial decomposition is shown, where there are two connections between the bound and the free block corresponding to the functions g_0 and g_1. The number of entries to individual blocks depends on the number of LUT block entries in which a given structure will be mapped.

In the classical approach, the sets X_b bound and X_f free do not contain common elements. In some cases, one of the variables contained in the bound set may play the role of a bound function. Such a variable is then attached to both the bound and the free set, which leads to a reduction in the number of necessary bound functions and thus the reduction of the complexity of the bound block responsible for the implementation of the function g. This approach is called non-disjoint decomposition (Dubrova, 2004; Dubrova *et al.*, 2004), and the methods of searching for appropriate variables are presented by Kubica and Kania (2017a) as well as

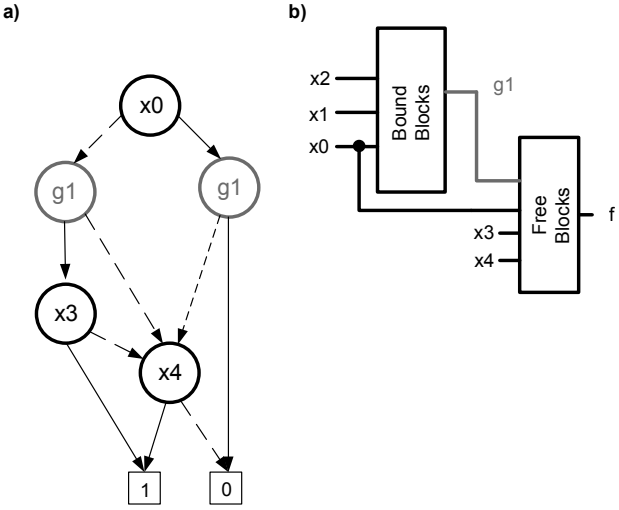


Fig. 2. Non-disjoint decomposition: BDD in which g_0 is replaced by x_0 (a), partition model for non-disjoint decomposition (b).

Kubica *et al.* (2021b). Not every variable can play the role of function g, and such variables often do not exist at all. In the case shown in Fig. 1, the variable x_0 performs the same role as the function g_0. Using non-disjoint decomposition, the BDD can be represented after decomposition, as shown in Fig. 2(a), leading to a partition corresponding to non-disjoint decomposition, as shown in Fig. 2(b).

The use of non-disjoint decomposition reduces the number of LUTs and the average fanout. The explanation of the above observation will be based on Figs. 1(c) and 2(b). The signals x_0, x_1, x_2 can be treated as

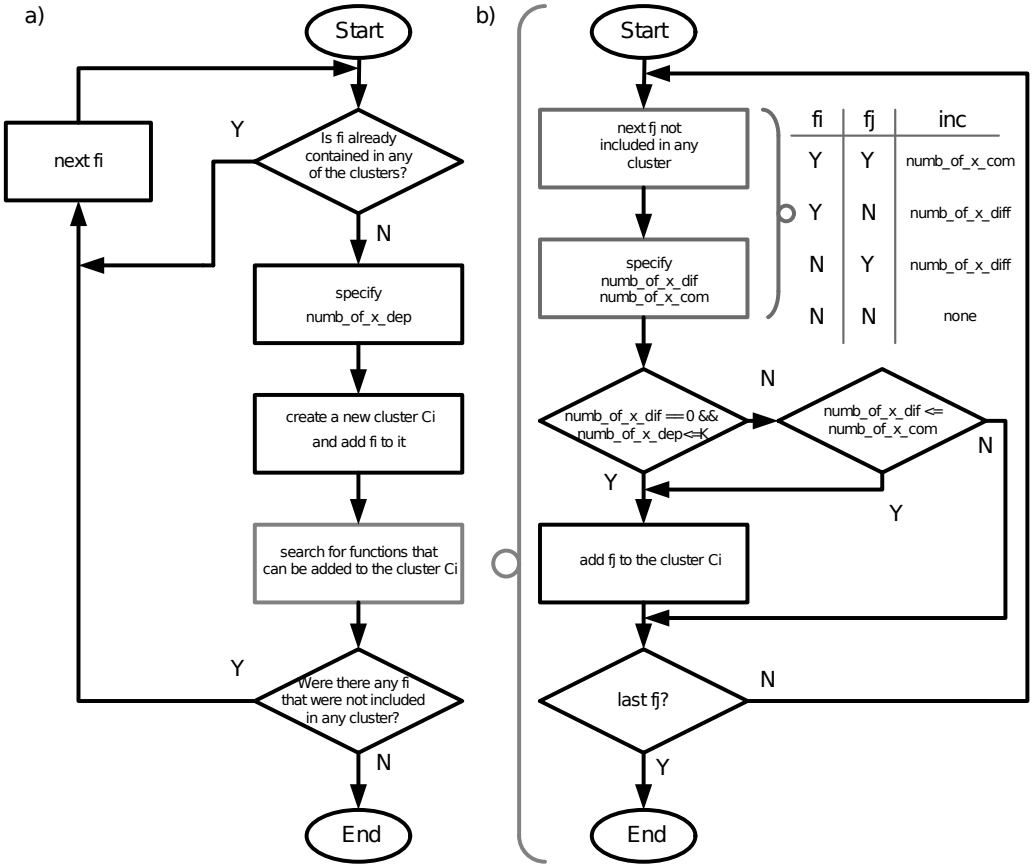


Fig. 3. Algorithms: initial division of a set of functions into clusters (a), adding functions to the cluster (b).

outputs of FPGA input buffers or outputs of LUTs from earlier decomposition steps. In the case of disjoint decomposition (Fig. 1(c)), the fanout of the signal x_0 is 2 (signal x_0 leads to two LUT3 blocks, g_0 and g_1), the x_1 fanout is 2, the x_2 fanout is 2. In the case of non-disjoint decomposition (Fig. 2(b)) the fanout of the signal x_0 is 2 (the signal x_0 goes to one LUT3 bound block and to the free block), but the fanout of x_1 is 1 and that of x_2 is 1. The number of LUT blocks has been reduced, as has the average fanout, which also reduces the dynamic power of the circuit.

Naturally, simple serial decomposition is the model upon which more complex decomposition models, such as iterative or multiple decomposition, are based. Search methods can be found in the works of Opara *et al.* (2018), Scholl (2001) or Kubica *et al.* (2021b; 2017). Additionally, apart from the methods using a single BDD cut, there are also those in which the number of cut lines in a single decomposition step may be greater than one (Kubica *et al.*, 2021b; Kubica and Kania, 2017b; 2016; 2019).

In engineering practice, combinational circuits are usually described by a multi-output function, not a single one, which indicates that it is necessary to extend the theory of decomposition to a set of functions in such a way as to obtain the most effective solutions after implementation. Section 6 proposes a new decomposition method aimed at reducing power consumption for the implementation of a multi-output function.

3. Initial selection of functions for the cluster

Appropriate mapping of a multi-output function can lead to the sharing of logic resources between structures associated with individual single functions. Naturally, this process does not always lead to solutions that are efficient in terms of both logic resources and power consumption. It thus becomes necessary to define which of the functions should be implemented together.

In the first steps of the synthesis before decomposition, there is a need to pre-group the functions. The entire set of functions is divided into clusters containing one or several functions that can potentially be implemented together.

The initial selection of functions for individual clusters is performed by defining a set of common dependent variables. Initially, the list of functions among

which suitable cluster candidates are searched contains $f_0, \ldots, f_{n-1}$. First, a cluster C_0 containing only f_0 is created, and then, a candidate most similar to f_0 is searched for among $f_1, \ldots, f_{n-1}$ and attached to C_0. The similarity is determined based on the number of common and different variables, and the process repeats in a loop. The essence of the initial division of a set of functions into clusters is shown in the algorithm in Fig. 3(a). In this algorithm, functions are selected that do not belong to any of the previously created clusters. Then, the number of variables on which the given function is dependent is determined ($numb\_of\_x\_dep$). A new cluster C_i is created for it, and then, attempts are made to join that cluster with other functions that were not included in the earlier clusters. This procedure has been marked in grey in this algorithm and is described in the flowchart shown in Fig. 3(b).

The algorithm for adding functions to the cluster C_i with initially added f_i (Fig. 3(b)) begins with determining the number of common variables ($numb\_of\_x\_com$) and that of different variables ($numb\_of\_x\_dif$) for functions not connected to any cluster f_j and the given function f_i. This procedure is presented in the form of a table describing the grey part of the algorithm in Fig. 3(b). If both functions depend on a given variable x (marked in the row of the table as Y, Y), the parameter $numb\_of\_x\_com$ is incremented. If one function is independent of the variable x (marked as N, Y or Y, N in the table), the parameter $numb\_of\_x\_dif$ is incremented. Otherwise, both parameters remain unchanged. This procedure looks for functions with the largest possible number of common variables and the smallest number of distinct variables. The function considered is included in the cluster if $numb\_of\_x\_dif = 0$ for the case where it is not necessary to decompose the given function (*numb_of_x_dep* $\leq k$) or if $numb\_of\_x\_dif \leq numb\_of\_x\_com$. The computational complexity of the algorithm is $O(n^3 \times m)$, where n is the number of functions and m is that of variables.

In the next stages of synthesis, individual clusters are subjected to further analysis, in which case decomposition and technology mapping algorithms are implemented. Unfortunately, the proposed decomposition algorithm is not always successful. In such a case, the cluster considered is divided into two (the set of functions included in the original cluster is partitioned). The newly created clusters (sets of functions) are subject to decomposition again. In the case of hard-to-decompose functions, this approach may lead to a situation where clusters split up into those that contain only a single function. If, in the case of single functions, the proposed algorithm is also ineffective, the only solution is to use decomposition associated with the Shannon expansion. Unfortunately, this type of decomposition is ineffective in terms of the use of logic resources and thus power consumption.

To decompose the multi-output function contained in the cluster, it becomes necessary to adequately describe this assembly with an appropriate form of the BDD.

4. Description of the multi-output function with the use of a PMTBDD

There are many methods of describing a multi-output function with a BDD. The most popular are the SBDD (Shared BDD) and the MTBDD (multi terminal BDD) (Minato, 1996; Hasan Babu and Sasao, 1999). In the case of the SBDD, single functions are associated with the individual roots of the diagram. In the case of the MTBDD, leaves are modified in such a way that they represent the output vectors (i.e., the values taken by the single functions included in the set). The number of such leaves may also be greater than 2.

From the perspective of decomposition (technology mapping), it is most often better to implement a multi-output function than single functions separately. Unfortunately, this is not always the case. Too many or poorly chosen functions in one set can cause a significant increase in the number of cut nodes and thus in that of bound functions, which means that for a multi-output function (MTBDD), decomposition depends on the selection of variables for the related set X_b and the cut level, and on the selection of functions included in the set. This process leads to the necessity of developing quick methods allowing attachment or detachment of a selected single function from a multi-output function. Thus, it is a difficult task for both the MTBDD and the SBDD. In this situation, the authors proposed the introduction of a new form of the PMTBDD, as presented by Opara *et al*. (2019) or Kubica *et al.* (2021b; 2017).

The essence of the representation of a multi-output function in the form of a PMTBDD is adding new variables represented by additional nodes. These nodes are associated with single functions included in the multi-output function. Thus, compared with the MTBDD, multi-bit leaves, which represent the values of the multioutput function, are replaced with subdiagrams that represent additional nodes. This approach allows us to easily attach or detach a single function using the well-known procedures bdd_or() and bdd_compose(), as shown by Opara *et al*. (2019) or Kubica *et al.* (2021b; 2017).

Consider the two functions described by PMTBDD, as shown in Fig. 4 (the function f_0 is represented by the diagram of Fig. 4(a), and function f_1 is represented by the diagram in Fig. 4(b)).

Considering the diagrams in Fig. 4, additional nodes associated with individual functions have been introduced (marked in grey). The rules for placing new nodes in PMTBDD are described by Opara *et al*. (2019) or Kubica *et al.* (2021b; 2017). Additionally, cut nodes in both

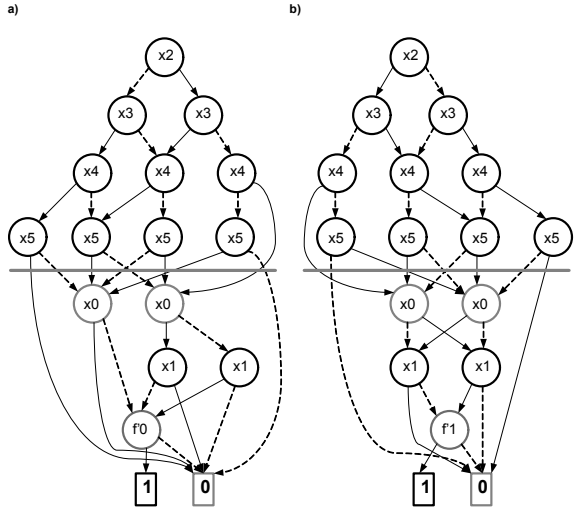


Fig. 4. PMTBDD representing the functions f_0 (a) and f_1 (b).

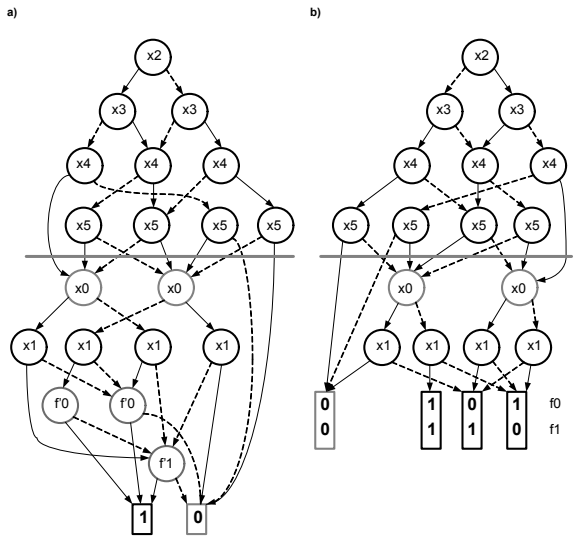


Fig. 5. Representation of the multi-output function $f_0 f_1$ in the form of a PMTBDD (a) and an MTBDD (b).

cases are marked in grey. As a result of merging both diagrams (bdd_or), a PMTBDD was created representing the $f_0 f_1$ functions, as shown in Fig. 5(a). As before, additional nodes and cut nodes are marked. Figure 5(b) shows the same multi-output function $f_0 f_1$ in the form of the MTBDD.

Naturally, a multi-output function described in both the form of an MTBDD and a PMTBDD can be decomposed by horizontal cutting. In the case under consideration, the diagrams describing the multioutput function from Fig. 5 have three cut nodes. Therefore, it is necessary to use two bound functions. Similarly, for the diagrams describing single functions (Fig. 4), the number of cut nodes for single functions is also three. Therefore, for function f_0, it is necessary to introduce two bound functions, and function f_1 requires two bound functions. Comparing the implementation of a multi-output function with of single functions, the implementation of the multi-output function leads to the reduction of the number of necessary bound functions from four to two. As a result, the bound block implementing them a smaller number of logic resources of the FPGA device and thus will require less power consumption.

The reduction of power consumption can also be performed in the process of decomposition itself. Unfortunately, this process requires the modification of the PMTBDD diagram to include additional information necessary for its reduction.

5. SWMTBDDs and SWPMTBDDs

With digital devices, the total power consumed by a given device P_{dev} consists of two components: the static power, P_{stat} and the dynamic power, P_{dyn}, which can be summarized by the relationship (1):

$$P_{dyn} = P_{stat} + P_{dyn}. \tag{1}$$

The static power P_{stat} is strictly dependent on the technology in which the given device was made. Static power is influenced by a number of factors, such as gate leakage, drain junction leakage, and subthreshold current. Due to the nature of static power, the minimization of its consumption using synthesis algorithms is limited. The situation is marginally different in the case of the dynamic power P_{dyn}. The value of dynamic power can be determined from

$$P_{dyn} = \frac{1}{2} V_{dd}^2 f \sum_{i=1}^{n} C_i SW_i. \tag{2}$$

Analyzing (2), the dynamic power P_{dyn} depends on a number of parameters, such as the supply voltage of the circuit V_{dd} or the operating frequency of the circuit f and the sum of parameters for n circuit nodes. With FPGA devices, these nodes are configurable logic blocks. The capacitance C_i must be reloaded in the event of a logical transition in a given node to the opposite state. It thus becomes critical to determine for each of i nodes how often the logic state changes to the opposite one. For this purpose, an additional parameter has been introduced: the switching activity for the node considered (SW_i). This parameter can be influenced at the stage of logic synthesis. Since the influence of the synthesis tools on the capacity of C_i is small, one can try to minimize the absorbed dynamic power by reducing the switching activity (Kubica *et al.*, 2021a).

The switching activity is directly related to the probability of a given variable's state transition. Thus, we can talk about the probability of transition from state 0

to 1 and from state 1 to 0. If all variables are mutually independent, the value of the switching activity can be described by

$$SW = 2P(x)(1 - P(x)), \tag{3}$$

where $P(x)$ is the probability of the value 1 for the variable x, and $(1 - P(x))$ is the probability of 0 for the same variable. When analyzing a particular system, we usually know nothing about the probability of 0 or 1 appearing on a given system input. In this situation, it becomes necessary to make some assumptions about these probabilities. Most often, we assume that the probability of the occurrence of the value 0 is the same as the probability of the occurrence of the value 1, therefore $P(x = 0) = P(x = 1) = 0.5$. Ferreira *et al.* (2000) present a model of power estimation with the use of a BDD. Conversely, in the work of Bogliolo *et al.* (1998), time relations were introduced into the BDD. The essence of determining SW for combinational circuits can be found in the work of Costa *et al.* (1997).

Classic BDD do not contain information on the probability P value and the switching activity SW. The usefulness of such diagrams in the process of minimizing dynamic power consumption is therefore limited. Kubica *et al.* (2021a) modify the classic BDD (ROBDD) in such a way that additional factors ($P(x)$ and SW) were placed at individual nodes. This approach made it possible to search for the appropriate decomposition that aimed to minimize dynamic power consumption depending on the parameters contained in the nodes. This new form of the BDD has been named the SWBDD (Kubica *et al.*, 2021a). Unfortunately, the SWBDD is a description of a single function; therefore, its usefulness for the implementation of a multi-output function is limited.

In this situation, the authors propose introducing two new types of diagrams describing multi-output functions while considering parameters related to power.

The first diagram proposed by the authors is the SWMTBDD, which is an extension of the MTBDD. For each of the nodes, additional parameters are determined related to the probability of obtaining the value 1 (P) for each of the functions from the set described in the SWMTBDD under consideration. Thus, the switching activity (SW) is determined using Eqn. (3), for each function. Figure 6 shows an example of an SWMTBDD obtained after determining additional parameters for the MTBDD from Fig. 5(b).

Because the multi-output function considered consists of two functions, f_0 and f_1, a pair of parameters (P, SW) for each of the functions were determined for each of the nodes. The parameters P were determined based on the analysis of the respective paths. The probability is calculated similarly to the Shannon expansion. For each BDD node associated with a variable

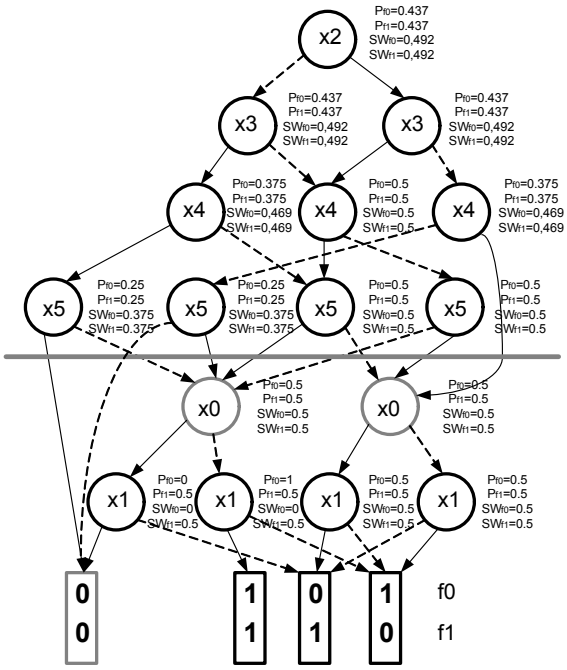


Fig. 6. SWMTBDD example.

x, the following formula is applied:

$$P(node) = P(x)P(node_{x_i=1}) + (1 - P(x))P(node_{x_i=0}), \tag{4}$$

where $node_{x_i=1}$ and $node_{x_i=0}$ are child nodes for true and false edges.

In the available BDD programming libraries, to unify the execution of all logical operations, the introduction of the ITE (if-then-else) operator was proposed (Minato, 1996). ITE is given by

$$ITE(x, y, z) = x \cdot y + \overline{x} \cdot z. \tag{5}$$

Because there is one operator, one result table (cache) is required. Basic logical operations can be expressed with a single operator, e.g., $f \cdot g = ITE(f, g, 0)$, $f + g = ITE(f, 1, g)$. The result of the operation can also be obtained iteratively similar to the Shannon expansion:

$$\begin{aligned} ITE(f, g, h) = ITE(x_i, \\ ITE(f_{x_i=1}, g_{x_i=1}, h_{x_i=1}), \\ ITE(f_{x_i=0}, g_{x_i=0}, h_{x_i=0})). \end{aligned} \tag{6}$$

Based on the expression (6), a typical ITE calculation algorithm is constructed (Algorithm 1, lines 7–8). To avoid calculating the results for the same arguments many times, the ResultTable is used (lines 3–4). In the proposed solution, each node is associated with an array of probabilities, and thus, the algorithm calculates probabilities if the result is not included in the result table

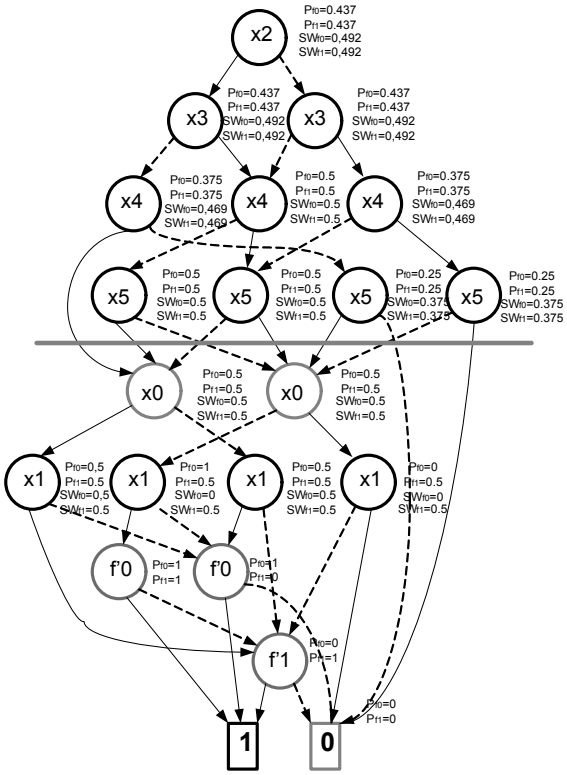


Fig. 7. SWPMTBDD example.

Algorithm 1. ITE operator.

Require: bdd I, bdd T, bdd E

1: **if** terminal case applies to I, T, E **then**
2: **return** computed result;
3: **else if** found in ResultTable **then**
4: **return** result from ResultTable;
5: **else**
6: x smallest index var from among roots of I, T, E;
7: $PosFactor = ITE(I_{x=1}, T_{x=1}, E_{x=1})$;
8: $NegFactor = ITE(I_{x=0}, T_{x=0}, E_{x=0})$;
9: $R = FindNodeInResultTable$
$(x, NegFactor, PosFactor)$;
10: **if** not found R **then**
11: $R.high = PosFactor$;
12: $R.low = NegFactor$;
13: **for** all functions f_i **do**
14: $R.P[f_i]$
$= P(x) \times PosFactor.P[f_i]$
$+(1 - P(x)) \times NegFactor.P[f_i]$;
15: **end for**
16: **end if**
17: InsertIntoResultTable((I,T,E), R);
18: **return** R ;
19: **end if**

(lines 13–15).

The second diagram proposed by the authors is SWPMTBDD, in which the same set of parameters was introduced for the nodes associated with the variables x. From the PMTBDD shown in Fig. 5(a), the SWPMTBDD can be derived as shown in Fig. 7.

The multi-output function described by the SWPMTBDD can be decomposed to minimize the dynamic power consumption.

The form of the PMTBDD and the SWPMTBDD allows us to conveniently combine single functions into diagrams of sets of functions or to separate them. This process was described in earlier works (Opara *et al.*, 2018; Kubica *et al.*, 2021b, 2017). During merging, the logic sum operator is used for the functions multiplied by the additionally introduced variables $f'_0 f'_1$ (Eqn. (7)) represented by the diagram in Fig. 5(a):

$$f_0 f'_0 + f_1 f'_1. \tag{7}$$

The standard operator ITE is used for a logic sum implementation in libraries for BDD operations. By making some modifications to the ITE algorithm, information about the probability can be retained.

When combining diagrams, we can use previously calculated probabilities for individual functions, thanks to which we can obtain an effective tool for combining (and separating) functions into appropriate sets. Combining functions into appropriate sets should be done in such a way that the number of cut nodes does not increase. Thus, it is possible to obtain the sharing of logic resources (Kubica *et al.*, 2021b) and the minimization of the number of logic blocks used. From the perspective of power minimization, it is critical to use additional information about the probability and the switching activity contained in the diagram.

6. Decomposition of the multi-output function aimed at minimizing dynamic power

The SWPMTBDD can be decomposed with the same methods as diagrams without probability information. By combining single function diagrams into SWPMTBDDs and then cutting off the top of the diagram, one can obtain a common bound block for n functions. Figure 8(a) shows a bound block with three outputs common for two functions, f_0 and f_1. If no efficient decomposition can be found for a particular multi-output function, the functions must be separated. Then, these functions should be decomposed separately. In this case, it is possible to search for sharing only for some functions of the bound block. Figure 8(b) shows that only a single function g_0 is shared. The search process is described by Kubica *et al.* (2021b). The essence of function decomposition with the use of a BDD is the cutting of the diagram, which leads

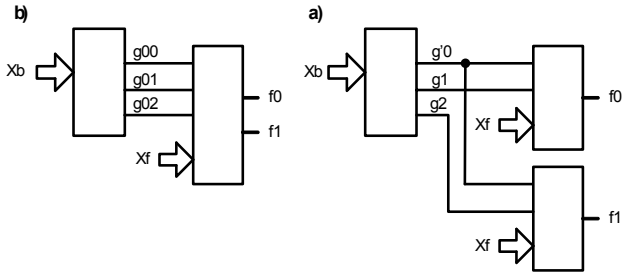


Fig. 8. Idea of sharing bound functions: all (a), selected (b).

to the fact that the variables above the cut line belong to the bound set; the critical determination is which variables to include in this set. Therefore, different orderings of variables in the BDD are analyzed. The effectiveness of the solution found should be assessed each time. For this purpose, the cost function was introduced, which depends on a number of parameters, such as the number of bound functions (associated with the complexity of the bound block), the cardinality of the bound set X_b or the switching activity. The idea of searching for an effective decomposition using the cost function is presented in Algorithm 2. This algorithm is a key element of the strategy described in this article further later.

The key element of the algorithm is the reordering of variables in the BDD (line 2). Because the decomposition is designed to minimize the dynamic power consumption, an important element of determining the cost function is considering the switching activity (lines 4–7). SW is evaluated based on the formula (3). The probability is calculated similarly to the Shannon expansion and formula (4). The order of variables in the diagram is changed by a series of swaps of adjacent nodes. The probabilities do not need to be recalculated for all nodes in the diagram when the order of the variables is changed; only swapped nodes must be recalculated. In the proposed solution, the sum SW is determined for the nodes above the cut line for all functions in the set. Next, non-disjoint decomposition is searched for. Shared bound functions are created. The next step is to calculate the cost of the solution found (lines 10–18). For this purpose, the sum of all g functions (which are associated with single LUTs) and that of the cardinality of bound sets are counted. The temporary numeric value of the cost function is determined in line 18. The expression uses bit shift operators to give the appropriate parts of the expression appropriate weights, which allows the cost value to be stored as a single number. The factor $num\_g - sum\_card\_X_b$ has the greatest weight and SW_cost_sum the least. Based on experiments, the SW_cost_sum is always smaller than 2^7; therefore, the shift is seven bits. In addition, num_ndisj is always less than 2^5; therefore, the offset is five bits. The last step is

Algorithm 2. Decomposition_step.

Require: multioutput_function
1: **for** n times **do**
2: change_BDD_var_order();
3: get_cut_nodes();
4: SW_cost_sum = 0;
5: **for** all functions f_i **do**
6: SW_cost_sum + = calculate_SW_activity_sum _of_nodes_over_cut_ln(f_i);
7: **end for**
8: find_nondisjoint_decomp();
9: generate_shared_g()
10: {Calculate cost }
11: num_ndisj = sum_nondisjoint_vars(); {number of nondisjoint g}
12: num_g = sum_shared_g(); {number of shared g }
13: sum_card_Xb =0; {sum of cardinalities of bound sets }
14: **for** all functions f_i **do**
15: num_g + = num_not_shared_g(f_i);
16: sum_card_Xb + = num_ dependend_variables(f_i);
17: **end for**
18: tmp_cost = ((num_g − sum_card_Xb)<<5 + num_ndisj)<<7 − SW_cost_sum);
19: **if** tmp_cost > best_cost **then**
20: best_cost = tmp_cost;
21: remember_solution();
22: **end if**
23: **end for**

to compare the (temporary) solution found with the best solution obtained thus far. If an interim solution is more effective, it is kept as the best one found thus far.

7. Synthesis strategy

The described methods lead to a consistent synthesis strategy named by the authors PowerDekBDD_MF. This strategy is a development of the DekBDD (Opara *et al.*, 2019) and PowerDek (Kubica *et al.*, 2021a) strategies implemented earlier by the authors. The DekBDD's strategy was intended to minimize the use of the number of logic resources for the implementation of the multi-output function. In contrast, the PowerDek strategy is geared toward minimizing power consumption for single functions.

Before applying the decomposition methods described in Section 6, there is a need to pre-classify functions into clusters using the algorithm presented in Section 3. Additionally, it is necessary to prearrange the order of variables. The next step is the implementation of the decomposition algorithm aimed at minimizing dynamic power, which is shown in Fig. 9.

The essence of this algorithm is the successive change in the order of the variables performed using the $swap(x_i, x_j, F)$ operation for the multi-output function F. For different orderings of variables, a non-disjoint decomposition (process of technology mapping optimization) is searched, and parameters related to power (P and SW) are determined. Next, the temporary cost parameter ($Costtemp$) is estimated, which is then compared with the best-known solution ($Cost$). Appropriate selection of functions to F is of key importance; therefore, the SWPMTBDD form is used, which allows easy grouping of functions.

It is possible that, after performing the decomposition with the algorithm in Fig. 9, we will not obtain a satisfactory result. In this situation, the operations indicated in Section 3 are performed, i.e., the division of the cluster or the implementation of decomposition associated with the Shannon expansion.

The presented algorithms were implemented using the PowerDek_MF tool, which was employed to perform the experiments.

8. Experimental results

To confirm the effectiveness of the proposed algorithms, multiple experiments were performed. The set of the described benchmarks (CBEAL, 2004) was used for the experiments. The benchmark circuits described by *.pla files were decomposed using the PowerdekBDD_MF tool. Three series of experiments were performed.

In the first series of experiments, two academic synthesis tools developed by the authors of the DekBDD (Opara *et al.*, 2018; Kubica *et al.*, 2021b; 2017) and PowardekBDD_MF were compared. The DekBDD strategy is the traditional serial decomposition performed by a single BDD cut aimed at minimizing the number of logic blocks. The PowerdekBDD_MF strategy is the development toward minimizing power consumption. The goal of this series of experiments was to compare the two approaches in terms of the number of LUT blocks (limited to blocks with five inputs—LUT_5), the number of logical levels and the switching activity. It is also critical to determine the effect of the number of shared bound functions and the number of non-disjoint decompositions found.

The results of the comparison are summarized in Table 1. The first three columns describe the benchmark (name, number of inputs and number of outputs). The remainder of the table is split into two parts associated with the systems being compared. Two cases can be distinguished for both parts: with a non-disjoint decomposition "$withnon-dis.dec.$" and without "$withoutnon-dis.dec.$" Table 1 lists the number of LUT blocks "LUT_5", the number of shared bound functions g "$Shared$", the number of logic levels "$Levels$", and the switching activity "SW". In addition, information about the synthesis time "$time$" (expressed in seconds) and the value of the fanout "$fanout$" are also provided. Additionally, in the section "$withnon-dis.dec.$" the number of non-disjoint decompositions "$Ndisj$" is given.

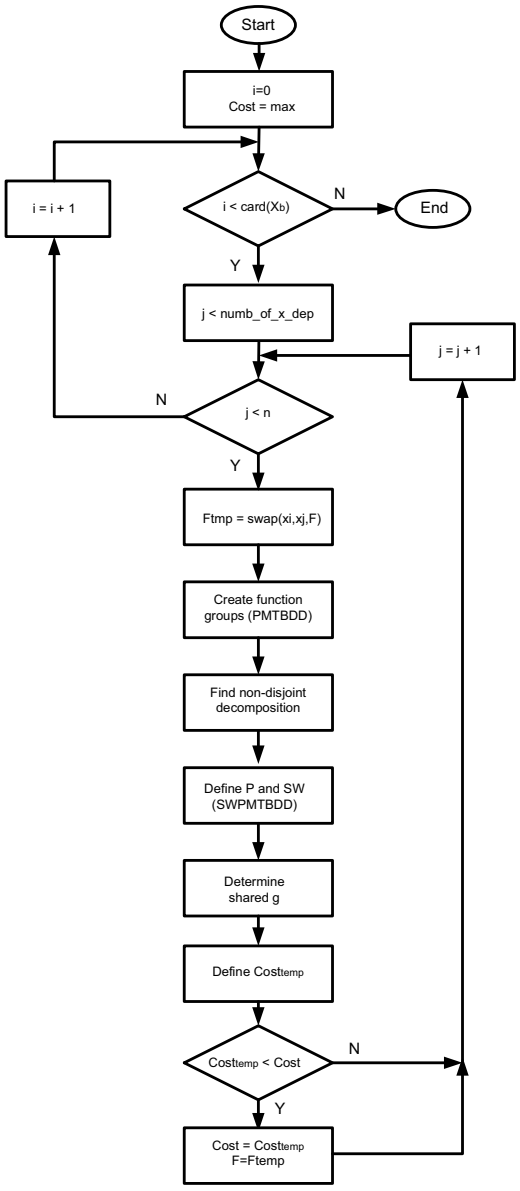


Fig. 9. Decomposition algorithm aimed at minimizing dynamic power.

At the bottom of Table 1, the individual sums are presented, which are summarized in the form of graphs in Fig. 10. To improve readability, non-disjoint decomposition is denoted as NDD.

Comparing the obtained results in terms of

Table 1. Comparison of the DekBDD and Powerdek_MF systems.

Benchm.	Inputs	Outputs	DekBDD													PowerdekBDD_MF												
			with non-dis. dec.							without non-dis. dec.						with non-dis. dec.							without non-dis. dec.					
			LUT_5	Ndisj	Shared	Levels	time	fanout	SW	LUT_5	Shared	Levels	time	fanout	SW	LUT_5	Ndisj	Shared	Levels	time	fanout	SW	LUT_5	Shared	Levels	time	fanout	SW
5xp1	7	10	13	3	1	2	0.01	3.1	6.00	13	4	2	0.01	3.1	5.99	13	3	1	2	0.01	3.1	6.00	13	4	2	0.01	3.1	5.99
9sym	9	1	6	0	0	3	0.01	2	2.77	6	0	3	0.01	2	2.77	6	0	0	3	0.01	2	2.77	6	0	3	0.01	2	2.77
alu2	10	8	25	6	2	3	0.01	3.49	10.42	26	1	3	0.01	3.611	9.85	26	9	2	4	0.02	3.611	11.00	26	1	3	0.04	3.611	9.68
alu4	14	8	376	47	7	17	0.28	4.41	167.76	642	14	18	0.35	4.441	282.25	446	50	0	15	0.37	4.407	196.91	711	13	17	0.65	4.459	299.24
apex7	49	37	130	103	27	8	0.34	3.59	45.86	198	37	7	0.39	3.891	68.87	94	115	25	8	0.52	3.224	32.31	306	44	13	0.73	4.118	108.76
b12	15	9	21	6	2	3	0.01	2.86	6.13	20	2	3	0.01	2.914	6.00	18	3	2	2	0.02	2.788	5.40	23	2	3	0.06	3.026	7.13
b9	16	5	44	28	9	5	0.03	2.47	14.14	72	23	6	0.05	3.115	23.33	45	25	9	5	0.05	2.477	14.33	66	14	7	0.04	3.065	22.03
bw	5	28	28	0	0	1	0.02	4.91	11.66	28	0	1	0.01	4.909	11.66	28	0	0	1	0.01	4.909	11.66	28	0	1	0.02	4.909	11.66
c8	28	18	30	15	3	4	0.02	2.47	13.24	39	5	4	0.02	2.806	17.27	29	14	3	4	0.03	2.439	12.36	40	5	4	0.03	2.838	17.31
cht	47	36	37	0	0	2	0.01	2.27	14.00	37	0	2	0.01	2.274	14.00	37	0	0	2	0.01	2.274	14.00	37	0	2	0.01	2.274	14.00
clip	9	5	18	16	4	3	0.03	3.52	7.73	17	5	3	0.04	3.231	7.12	18	16	4	4	0.03	3.519	8.10	17	5	3	0.04	3.231	7.12
cm162a	14	5	11	2	4	3	0.01	2.12	4.24	12	5	3	0.01	2.231	4.04	11	2	4	3	0.01	2.12	4.24	12	5	3	0.01	2.231	3.98
cm163a	16	5	9	2	2	3	0.01	1.8	2.98	10	2	3	0.01	1.923	3.26	9	2	2	3	0.01	1.8	3.28	10	2	3	0.01	1.923	3.26
cm85a	11	3	8	1	2	3	0.01	1.89	3.10	9	4	3	0.01	2.2	2.69	8	1	2	3	0.02	1.895	2.99	9	4	3	0.01	2.2	2.69
con1	7	2	3	1	0	2	0.005	1.5	1.40	4	0	2	0.01	1.818	1.64	3	1	0	2	0.01	1.5	1.40	4	0	2	0.005	1.818	1.83
count	35	16	44	35	12	6	0.07	2.72	19.96	49	15	5	0.08	2.881	17.17	42	41	9	6	0.1	2.675	18.53	46	14	6	0.09	2.765	15.70
duke2	22	29	265	122	18	10	0.57	4.45	87.53	408	37	12	0.67	4.507	122.96	251	124	17	10	0.69	4.44	72.62	382	39	11	0.85	4.525	97.98
e64	65	64	106	0	37	8	1.29	2.9	2.11	106	37	8	1.25	2.901	2.11	106	0	37	8	1.75	2.901	2.06	106	37	8	1.71	2.901	2.06
example2	85	66	124	88	22	6	0.34	2.83	31.20	152	29	5	0.34	2.987	38.62	123	90	21	6	0.41	2.788	31.75	142	25	6	0.41	3.075	33.15
f51m	8	8	10	2	1	3	0.01	2.5	4.94	12	2	3	0.01	2.8	5.65	10	2	1	3	0.01	2.5	4.94	12	2	3	0.01	2.8	5.65
inc	7	9	20	3	0	3	0.01	3.52	8.08	20	0	2	0.01	3.593	7.97	20	3	0	3	0.01	3.519	7.89	20	0	2	0.01	3.593	7.32
misex1	8	7	15	0	2	2	0.01	3.13	5.66	15	2	2	0.01	3.13	5.66	15	0	2	2	0.01	3.13	5.66	15	2	2	0.01	3.13	5.66
misex2	25	18	33	10	4	3	0.03	2.78	6.22	37	10	3	0.03	2.903	5.15	32	12	5	4	0.03	2.772	4.56	37	8	3	0.02	2.855	4.02
misex3	14	14	434	96	14	14	0.52	4.54	170.12	497	18	12	0.68	4.54	181.26	297	84	12	11	0.57	4.514	109.25	473	16	11	0.82	4.495	159.97
misex3c	14	14	134	36	1	8	0.14	4.27	57.97	185	7	10	0.22	4.417	76.56	125	42	1	10	0.17	4.324	54.27	177	4	9	0.3	4.393	73.90
mux	21	1	21	17	0	6	0.02	2.48	9.00	53	0	9	0.02	3.324	24.26	17	16	0	7	0.02	2.211	7.46	39	0	9	0.03	3.133	15.35
pcle	19	9	18	8	5	4	0.03	2.38	5.27	21	4	4	0.02	2.6	8.40	18	11	3	4	0.02	2.405	5.22	21	4	4	0.02	2.55	6.13
pdc	16	40	117	42	7	6	0.39	4.29	34.40	137	15	6	0.39	4.412	36.48	120	51	7	8	0.4	4.287	35.63	160	14	8	0.44	4.386	46.72
rd53	5	3	3	0	0	1	0.01	2.25	1.27	3	0	1	0.005	2.25	1.27	3	0	0	1	0.01	2.25	1.27	3	0	1	0.01	2.25	1.27
rd73	7	3	6	0	2	2	0.01	2.23	2.97	6	2	2	0.01	2.231	2.97	6	0	2	2	0.01	2.231	2.97	6	2	2	0.01	2.231	2.97
rd84	8	4	9	1	2	3	0.02	2.65	3.32	10	2	3	0.01	2.778	3.62	9	1	2	3	0.01	2.647	3.32	10	2	3	0.02	2.778	3.67
sao2	10	4	33	14	4	5	0.03	3.84	10.65	33	8	5	0.03	3.814	9.20	33	12	6	6	0.04	3.837	7.99	31	7	5	0.03	3.805	8.44
sct	19	15	20	14	4	4	0.02	2.56	6.07	23	7	3	0.01	2.762	7.13	20	14	4	4	0.02	2.564	6.04	23	7	3	0.02	2.762	7.14
spla	16	46	372	133	32	12	0.99	4.61	126.67	467	54	13	1.33	4.638	154.08	328	123	36	11	1.28	4.587	100.34	439	48	14	2.56	4.626	140.66
squar5	5	8	8	0	0	1	0.01	3.23	3.49	8	0	1	0.01	3.231	3.49	8	0	0	1	0.01	3.231	3.49	8	0	1	0.01	3.231	3.49
t481	16	1	5	2	0	3	0.04	1.1	2.12	10	0	4	0.03	1.962	3.95	5	2	0	4	0.04	1.095	2.12	8	0	4	0.07	1.625	3.61
table3	14	14	772	90	9	13	0.65	4.43	306.82	1078	23	15	0.78	4.469	419.77	813	117	9	15	0.96	4.47	310.46	1187	42	17	1.33	4.544	446.61
term1	34	10	30	51	12	6	0.08	2.34	7.89	66	19	5	0.09	3.3	20.03	30	32	14	5	0.14	2.313	7.78	138	28	14	0.28	3.721	46.20
ttt2	24	21	40	16	8	3	0.03	2.97	14.26	45	13	3	0.03	3.203	16.86	40	16	8	3	0.04	2.969	14.14	46	12	3	0.05	3.214	17.27
vg2	25	8	72	46	5	6	0.11	3.68	28.75	77	13	6	0.13	3.755	25.24	65	43	9	7	0.17	3.567	26.22	394	15	17	0.71	4.32	169.94
x2	10	7	15	6	3	3	0.01	2.72	5.47	20	4	3	0.01	3.1	6.79	15	6	3	3	0.01	2.72	4.64	20	4	4	0.02	3.1	6.85
x4	94	71	109	41	18	4	0.24	2.67	37.22	124	25	4	0.28	2.844	42.99	113	51	18	5	0.31	2.696	37.28	124	23	4	0.41	2.775	44.10
Sum:			**3594**	**1103**	**285**	**207**	**6.485**	**126**	**1310.85**	**4795**	**448**	**212**	**7.435**	**133.8**	**1710.39**	**3455**	**1134**	**280**	**213**	**8.37**	**125.7**	**1214.64**	**5375**	**454**	**243**	**11.925**	**134.4**	**1893.30**
Geometric mean:			**31.45**			**3.92**	**0.04**	**2.87**	**10.93**	**37.93**		**3.93**	**0.04**	**3.08**	**12.72**	**30.57**			**4.06**	**0.05**	**2.85**	**10.35**	**40.07**		**4.28**	**0.06**	**3.08**	**13.27**

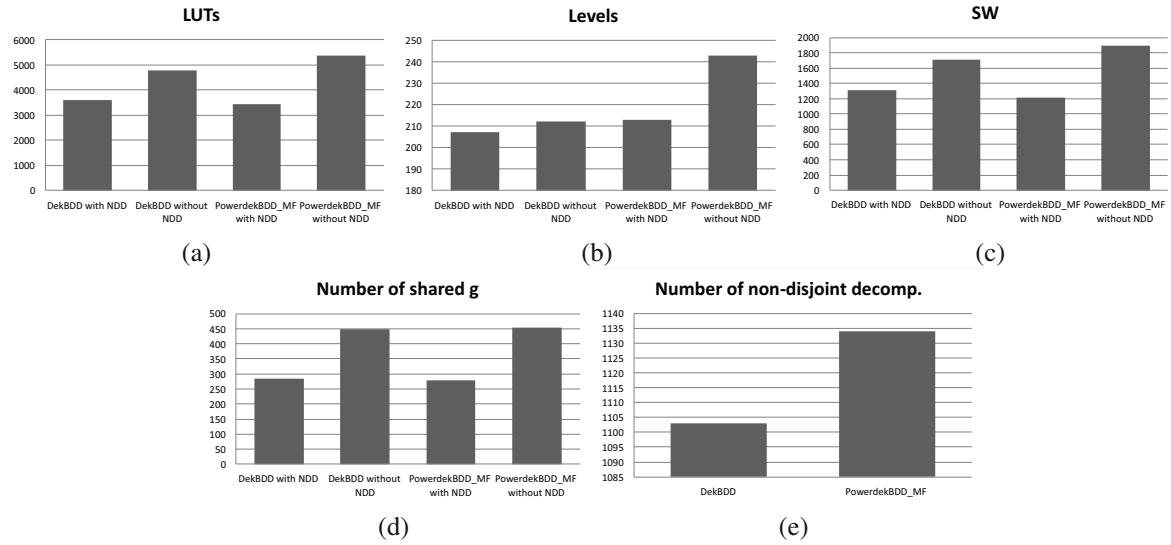


Fig. 10. Comparison of the DekBDD and PowerdekBDD systems in terms of: number of LUT blocks (a), number of logical levels (b), switching activity (c), number of shared bound functions (d), number of non-disjoint decompositions found (e).

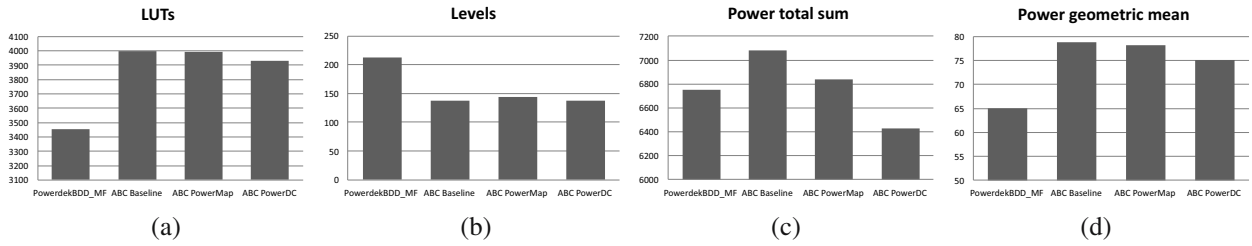


Fig. 11. Comparison of the PowerdekBDD_MF and ABC systems in terms of: number of LUT blocks (a), number of logical levels (b), power total sum (c) and geometric average (d).

the number of logic blocks (Fig. 10a), non-disjoint decomposition is shown to have strong impact (Opara and Kubica, 2018; 2017; Opara and Kania, 2009). In both systems, non-disjoint decomposition led to a significant reduction in the number of necessary LUTs compared with the exclusively disjoint decomposition. Additionally, considering the non-disjoint decomposition for the PowerdekBDD_MF system, a marginally lower number of blocks was obtained than for the DekBDD. Thus, the average fanout was reduced in every case where the non-disjoint decomposition was applied. This observation emphasizes the importance of the non-disjoint decomposition in methods aimed at minimizing power. Comparing the number of logical levels (Fig. 10(b)), we can see the advantage of the DekBDD system over PowerdekBDD_MF. It is essential to compare both systems in terms of the switching activity (Fig. 10(c)). As expected, the best results were obtained for the PowerdekBDD_MF system with the non-disjoint decomposition.

From the perspective of the implementation of the multioutput function, the number of shared bound functions is important. The obtained results for both systems are practically the same (Fig. 10(d)). In relation to the results for single functions presented by Kubica *et al*. (2021a) without shared resources, some observations can be made regarding the number of g functions. In Table 1, the given LUT5 is the sum of the number of non-shared and shared g functions ($LUT5_{multioutput} = g_{notshared} + g_{shared}$). Comparing LUT5 with the number of LUT blocks for single functions, the approximate dependence of $LUT5_{singleoutput} \approx LUT5_{multioutput} + g_{shared} = g_{notshared} + 2 \times g_{shared}$ in many cases is observed, which suggests that g functions are shared between two functions in the cluster in many cases.

Comparing the DekBDD and PowerdekBDD_MF in terms of the number of non-disjoint decompositions found (Fig. 10(e)), the PowerdekBDD_MF system is shown to be marginally more efficient.

The second series of experiments compared the PowerdekBDD_MF system with non-disjoint decomposition, for which the best results were obtained with the leading academic synthesis system ABC (BLSG, 2005). The method of performing synthesis by ABC must be specified each time by an appropriate set of commands (script). Depending on the script used, the

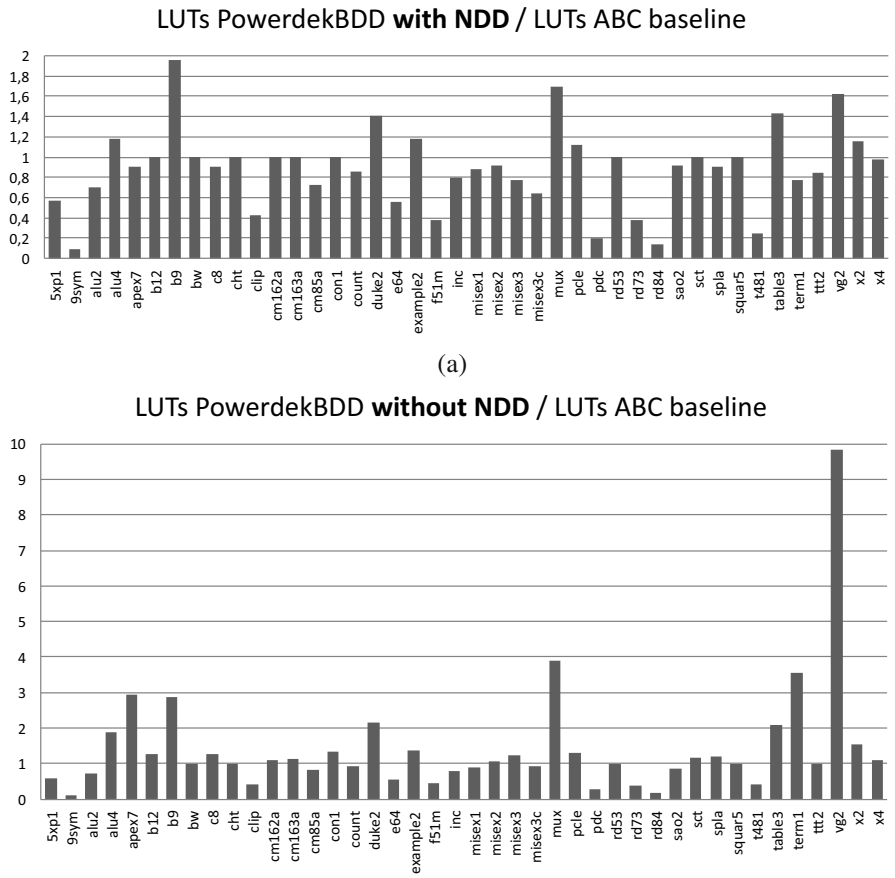
Table 2. Comparison of the PowerdekBDD_MF system with the ABC system.

Benchm.	Inputs	Outputs	PowerdekBDD_MF (with non-disj. dec.)			ABC Baseline			ABC PowerMap			ABC PowerDC		
			LUT_5	Levels	Power	LUT_5	Levels	Power	LUT_5	Levels	Power	LUT_5	Levels	Power
5xp1	7	10	13	2	30,80	23	3	47,86	27	3	52,25	21	3	40,98
9sym	9	1	6	3	14,56	67	5	131,85	68	5	132,04	68	5	128,43
alu2	10	8	26	4	54,55	37	3	70,57	39	3	74,35	37	3	67,57
alu4	14	8	446	15	925,03	377	6	692,28	375	6	678,96	376	6	640,96
apex7	49	37	94	8	178,08	104	4	189,59	93	4	173,24	109	4	198,51
b12	15	9	18	2	40,27	18	2	39,04	19	2	38,21	18	2	36,58
b9	16	5	45	5	86,12	23	3	47,57	25	4	49,05	23	3	47,2
bw	5	28	28	1	77,17	28	1	77,82	28	1	77,82	28	1	77,82
c8	28	18	29	4	65,51	32	3	65,91	33	3	63,33	32	3	61,73
cht	47	36	37	2	92,59	37	2	92,59	37	2	92,97	37	2	92,59
clip	9	5	18	4	43,14	42	3	81,43	37	4	71,36	42	3	69,43
cm162a	14	5	11	3	21,74	11	3	21,13	11	3	21,13	11	3	21,13
cm163a	16	5	9	3	19,49	9	2	21,38	9	2	19,71	9	2	19,71
cm85a	11	3	8	3	16,63	11	2	23,73	11	3	22,31	11	2	23,71
con1	7	2	3	2	7,31	3	2	7,35	3	2	7,35	3	2	7,35
count	35	16	42	6	90,36	49	3	83,07	51	3	83,3	49	3	78,8
duke2	22	29	251	10	443,89	178	4	294,81	173	4	289,06	178	4	280,89
e64	65	64	106	8	162,57	191	4	221,76	180	4	205,19	192	4	224,58
example2	85	66	123	6	231,71	104	3	176,72	104	3	166,23	104	3	169,82
f51m	8	8	10	3	22,73	26	3	56,56	36	4	73,83	25	3	52,38
inc	7	9	20	3	44,56	25	3	53,26	28	3	58,67	25	3	52,04
misex1	8	7	15	2	32,54	17	2	34,89	19	2	36,57	17	2	34,23
misex2	25	18	32	4	63,97	35	3	64,71	36	3	58,75	35	3	63,95
misex3	14	14	297	11	584,78	382	6	668,89	371	6	636,49	383	6	608,73
misex3c	14	14	125	10	272,88	194	5	358,97	196	5	349,67	193	5	343,3
mux	21	1	17	7	40,72	10	3	24,26	13	3	25,29	11	3	24,64
pcle	19	9	18	4	36,04	16	3	31,94	16	3	29,29	16	3	29,82
pdc	16	40	120	8	241,88	596	6	1082,39	601	6	1007,31	535	6	797,69
rd53	5	3	3	1	8,60	3	1	8,6	3	1	8,6	3	1	8,6
rd73	7	3	6	2	14,34	16	3	38,91	23	4	46,63	16	3	38,89
rd84	8	4	9	3	21,05	63	4	122,25	79	5	140,76	62	4	115,55
sao2	10	4	33	6	56,56	36	3	72,55	40	3	72,23	36	3	68,89
sct	19	15	20	4	44,98	20	2	37,93	20	2	37,44	20	2	37,93
spla	16	46	328	11	634,34	363	5	624,5	355	6	588,81	353	5	539,48
squar5	5	8	8	1	20,18	8	1	20,18	8	1	20,18	8	1	20,18
t481	16	1	5	4	10,23	20	4	31,05	19	4	29,72	18	4	27,24
table3	14	14	813	15	1467,82	568	6	894,72	560	6	846,9	569	6	833,05
term1	34	10	30	5	52,78	39	4	77,98	37	4	71,07	40	4	68,79
ttt2	24	21	40	3	82,52	47	3	88,68	40	3	80	47	3	85,05
vg2	25	8	65	7	134,68	40	4	66,06	42	4	66,02	42	4	66,24
x2	10	7	15	3	29,95	13	2	28,36	14	2	28,88	13	2	28,36
x4	94	71	113	5	234,38	115	3	208,6	115	3	204,87	114	3	198,26
Sum:			**3455**	**213**	**6754,03**	**3996**	**137**	**7082,7**	**3994**	**144**	**6835,84**	**3929**	**137**	**6431,08**
Geometric mean:			**30,57**	**4,06**	**64,98**	**39,80**	**2,98**	**78,85**	**41,20**	**3,13**	**78,19**	**39,65**	**2,98**	**75,07**

ABC system generates different synthesis results. For comparison, three synthesis scripts described by Chung and Brayton (2009) were used: Baseline (strash; dch; if -K 5 -e; ps -p;), PowerMap (strash; dch; resyn2; if -K 5 -p; ps -p;), and PowerDC (strash; dch; if -K 5 -p; mfs -p; ps -p;). The relevant results are summarized in Table 2. The first three columns describe the benchmark (name, number of inputs and number of outputs). The remainder of the table is divided into four parts associated with the compared systems (synthesis scripts). The comparison was made in terms of the number of LUT blocks with five inputs "LUT_5", the number of logic levels "$Levels$" and power. For this parameter, the value reported by ABC for the output circuit network is given.

At the bottom of Table 2, individual sums are presented, which are summarized in the form of graphs in Fig. 11.

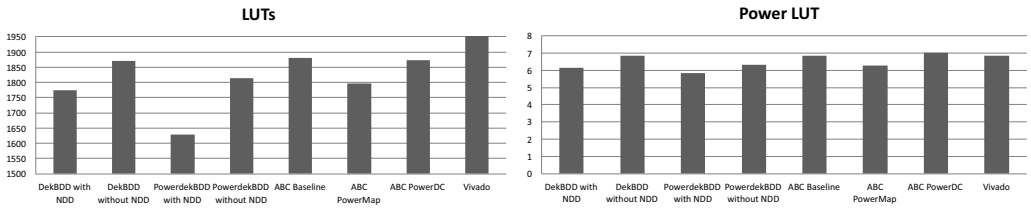
Comparing the two systems in terms of the number of required LUTs, a clear advantage of the PowerdekBDD_MF system (Fig. 11(a)). Unfortunately, this advantage comes at the expense of the dynamic properties of the circuit, as shown in Fig. 11(b), where the number of logic levels for the PowerdekBDD_MF system is much greater. It is difficult to draw far-reaching



(a)

(b)

Fig. 12. Ratio of the number of LUTs for PowerDek_MF to the ABC baseline: with (a) and without (b) non-disjoint decomposition, ordered ascending by circuit size.



(a) (b)

Fig. 13. Vivado synthesis results: number of LUT blocks (a), dynamic power for LUT (b).

conclusions in the case of the diagram in Fig. 11(c). The total power for PowerdekBDD_MF is smaller than that for ABC Baseline and ABC PowerMap but greater than that for ABC PowerDC. Looking at the geometric mean of power, PowerdekBDD_MF has the lowest value. We also decided to perform the third series of experiments, in which the descriptions of the decomposed systems were further analyzed (synthesized) with commercial tools to unequivocally define dynamic power consumption, which will ensure the reliability of the comparison.

To report the obtained results more clearly, Fig. 12 shows the ratio of the number of LUT blocks for the PowerdekBDD_MF system to the ABC baseline for each benchmark. This relationship is presented for two cases: with consideration of non-disjoint decomposition (Fig. 12a) and without it (Fig. 12(b)). The benchmarks are arranged in ascending order by the size of the circuit for the ABC baseline, and no regularity is shown based on the size of the circuit.

In the third series of experiments, the synthesis results obtained with the commercial Xilinx Vivado synthesis tool (Xilinx, 2021) were compared. Using academic tools, descriptions of the decomposed circuits in Verilog HDL were generated and then further

Table 3. Results obtained after synthesis in the Vivado tool.

Benchm.	Inputs	Outputs	DekBDD				PowerdekBDD				ABC Baseline		ABC PowerMap		ABC PowerDC		Vivado	
			with non-dis. dec.		without non-dis. dec		with non-dis. dec.		without non-dis. dec									
			LUT	Power LUT	LUT	Power LUT	LUT	Power LUT	LUT	Power LUT	LUT	Power LUT	LUT	Power LUT	LUT	Power LUT	LUT	Power LUT
5xp1	7	10	3	020. 7	6	02054	3	020. 7	6	02054	14	0205	14	020. 3	11	0205	14	0205
39sy	3	1	.	020.	.	020.	.	020.	.	020.	m	0204a	m	0204a	m	0204a	m	0204.
l u84	10	6	1.	02057	15	0207.	1.	02056	1.	02056	44	02064	40	02073	13	020m7	41	02106m
l u8.	1.	6	47.	023. 6	aa4	12 mm	174	0277.	4m	02311	44.	1204	166	02643	41a	02353	4m7	02346
l pex7	. 3	a7	5m	0244.	55	0244	57	02447	57	0244m	51	0241	50	02406	57	02445	5m	02445
b14	15	3	1a	020. 7	1m	02054	1a	020. m	1.	0205	14	020. .	1a	020. m	1a	020. a	1.	0205
b3	1m	5	43	02103	46	02114	43	02111	43	02111	1.	020m	15	020m6	1.	020m	4m	02103
bw	5	46	1.	02035	1.	02035	1.	02035	1.	02035	1.	0203m	1.	0203m	1.	0203m	1.	02035
c6	46	16	16	0207a	16	0207.	16	0207a	16	0207a	40	02073	40	0206a	41	02064	16	02074
cht	. 7	am	47	02106	47	02106	47	02106	47	02106	4m	02103	4m	02103	4m	02103	4m	02103
cup	3	5	15	02077	3	02054	1.	02071	3	02054	4m	021a6	1m	0206m	16	0206.	47	021a
cy 1m4l	1.	5	6	02045	6	02043	6	02046	6	02046	7	0204m	6	0204m	7	0204m	7	0204.
cy 1ma l	1m	5	m	02041	m	02041	m	02041	m	02041	m	02041	7	0204a	m	02041	m	0204
cy 65l	11	a	m	02041	m	02041	m	02044	m	02041	m	02041	m	02041	m	02041	m	02044
con1	7	4	4	02007	4	02006	4	02007	4	02007	4	02007	4	02007	4	02007	4	02007
co8nt	a5	1m	45	02075	45	02075	45	02075	45	02075	4.	0207m	a0	0203m	47	02065	45	0207m
d8ke4	44	43	3m	0246.	107	02a. 4	37	02466	37	02467	10a	02a4a	101	02a44	104	02a45	10.	02a4m
em	m5	m	63	024a5	63	024a5	30	024a3	30	024a3	1a.	02a4.	1am	02aa1	1. 5	02am3	1. a	02a5
exl y pue4	65	mm	mm	02445	70	024a.	m3	024a4	m6	02444	m	02416	m7	024a.	ma	02413	61	02a1
f51y	6	6	7	020.	7	020. 1	7	020.	7	020. 1	10	020.	10	020. .	10	020.	10	020. 6
inc	7	3	3	020a3	3	020a6	3	020a3	3	020a3	1a	020. 5	1a	020. 5	14	020. a	14	020. .
y i9ex1	6	7	6	020aa	6	020aa	6	020aa	6	020aa	6	020aa	6	020aa	6	020aa	6	020aa
y i9ex4	45	16	44	0206.	4.	02075	44	02061	4a	02061	4.	02076	45	0206	45	0206	45	02077
y i9exa	1.	1.	157	02574	1. 7	025a7	1a.	02 7a	13.	027. 7	150	0257m	1a5	02 34	1m0	02564	163	027.
y i9exac	1.	1.	6a	02a4.	106	02 1	6m	02a4m	65	02a4	107	02 1m	107	02 41	100	02a77	107	02 1m
y 8x	41	1	5	02044	5	02044	m	02045	m	0204m	5	02044	5	02041	5	02041	m	0204m
pcue	13	3	14	020. a	14	020. a	14	020. a	14	020. a	14	020. a	11	020a3	10	020a3	14	020. 4
pdc	1m	. 0	m5	024a3	mm	024. 5	m	024am	ma	024aa	1a3	02 a6	1a4	02 44	1a7	02 . a	113	02a63
rd5a	5	a	4	02014	4	02014	4	02014	4	02014	4	0201a	4	02014	4	0201a	4	02014
rd7a	7	a	.	020aa	.	020aa	.	020aa	.	020aa	m	02043	m	02043	m	02043	m	020a
rd6.	6	.	5	020. 5	5	020. 4	5	020. 5	5	020. 5	11	0205m	11	0205m	11	025m	11	02057
9l o4	10	.	13	0211m	16	02116	40	02101	40	0210.	40	020m4	40	020m7	40	02071	a6	021aa
9ct	13	15	14	0205a	1m	020m	14	0205a	1m	020m4	14	0205	14	0205	14	0205	1a	02051
9pul	1m	. m	1. 1	02 mm	1a3	02 ma	1a3	02 m5	1. 3	02 63	1a3	02 57	1a.	02 a4	1a5	02 4.	143	02 0m
9q8l r5	5	6	.	0204m	.	0204m	.	0204m	.	0204m	.	02045	.	0204m	.	02045	.	0204m
t. 61	1m	1	5	02013	5	02017	5	02013	7	02013	5	02016	5	02016	7	02016	5	02047
tl bıea	1.	1.	463	02m7a	465	02m64	4m4	02m 6	473	02m53	46m	02307	471	02m 5	477	02m77	4m0	02m4m
tery 1	a.	10	44	02064	a1	02114	16	0207m	43	02107	4m	02035	44	0206a	44	02076	43	0211.
ttt4	4.	41	44	02036	4m	0211m	45	02104	4.	02117	45	02101	4a	0203.	46	0211.	a0	0211m
vg4	45	6	46	0210m	46	0210m	46	0210m	4.	0203.	44	0206	13	020m6	4m	02036	46	02106
x4	10	7	7	020a4	3	020a.	7	020a.	3	020am	6	020a5	6	020a5	6	020a5	6	020a4
x.	3.	71	75	024mm	7.	024m4	75	024m	75	024m4	75	02a17	76	02a17	73	02a03	75	024ma
Sum:			**1774**	**6.141**	**1871**	**6.837**	**1628**	**5.842**	**1814**	**6.304**	**1882**	**6.867**	**1798**	**6.265**	**1874**	**7.034**	**1987**	**6.829**
Geometric mean:			**18.13**	**0.077**	**18.66**	**0.079**	**17.90**	**0.076**	**18.60**	**0.078**	**20.00**	**0.079**	**19.66**	**0.077**	**19.99**	**0.082**	**21.27**	**0.083**

synthesized in Vivado. Results were obtained regarding the number of LUT blocks used "LUT" and the value of dynamic power consumed by the created logic structures "$PowerLUT$", which are shown in Table 3. In this table, the first three columns are the description of the benchmark (name, number of inputs and number of outputs), and the remainder of the table is divided into five parts associated with the DekBDD, PowerdekBDD_MF,

ABC Baseline, ABC PowerMap and ABC PowerDC. In addition, the fragments of Table 3 containing the results for DekBDD and PowerdekBDD_MF systems are divided into two parts: with non-disjoint decomposition "$withnon-dis.dec.$" and without non-disjoint decomposition "$withoutnon-dis.dec.$" Additionally, there is a column presenting the results of the synthesis performed only in the Vivado system.

At the bottom of Table 3, individual sums are presented, which are summarized in the form of graphs in Fig. 13. To improve readability, the non-disjoint decomposition was denoted as NDD.

Analyzing the graph shown in Fig. 13(a), the best results in terms of the use of LUTs are obtained for the PowerdekBDD_MF system, which looks for non-disjoint decompositions. DekBDD also provides good results with regard to non-disjoint decomposition. However, it is critical to compare the systems in terms of the dynamic power consumption in the obtained LUT structures, which are shown in Fig. 13(b). Additionally, in this study, the PowerdekBDD_MF system with non-disjoint decomposition achieved the best results, which confirms the effectiveness of the developed methods in the process of reducing dynamic power.

9. Conclusions

The results of this study show that the proposed decomposition techniques related to the reduction of the switching activity can reduce dynamic power consumption. The approach that uses effective sharing of logic resources between the structures associated with the relevant functions included in the implemented team is effective both in terms of cutting down the use of logic resources and reducing dynamic power consumption. Results thus highlight the key impact of non-disjoint decomposition on decomposition efficiency.

Unfortunately, the proposed methods can only be used for relatively small circuits. With larger systems, the proposed solutions can be employed locally in a larger logical network. The scalability problem is a key flaw of the proposed approach, which we are still working on.

The proposed approach does not use the configuration capabilities of modern logic blocks contained in FPGA devices. This topic should be investigated in future research in the field of synthesis to reduce power consumption.

Acknowledgment

This publication was supported by the Department of Digital Systems (Rau12) and the Department of Computer Graphics, Vision and Digital Systems (Rau6) of the Silesian University of Technology under the 2023 statutory research project.

References

Akers, S.B. (1978). Binary decision diagrams, *IEEE Transactions on Computers* **27**(06): 509–516.

Ali, H. and Al-Hashimi, B.M. (2007). Architecture level power-performance tradeoffs for pipelined designs, *IEEE International Symposium on Circuits and Systems, New Orleans, USA*, pp. 1791–1794.

Ashenhurst, R.L. (1957). The decomposition of switching functions, *Proceedings of the International Symposium on the Theory of Switching, Cambridge, USA*, pp. 76–116.

Balasubramanian, P. and Anantha, K. (2007). Power and delay optimized graph representation for combinational logic circuits, *Engineering and Technology International Journal of Structural and Construction Engineering* **1**(8): 2481–2487.

Bard, S. and Rafla, N.I. (2008). Reducing power consumption in FPGAs by pipelining, *51st Midwest Symposium on Circuits and Systems, Knoxville, USA*, pp. 173–176.

Barkalov, A., Titarenko, L. and Chmielewski, S. (2020b). Improving characteristics of LUT-based Moore FSMSs, *IEEE Access* **8**: 155306–155318, DOI: 10.1109/ACCESS.2020.3006732..

Barkalov, A., Titarenko, L. and Mazurkiewicz, M. (2022). Improving the LUT count for Mealy FSMs with transformation of output collections, *International Journal of Applied Mathematics and Computer Science* **32**(3): 479–494, DOI: 10.34768/amcs-2022-0035.

Barkalov, A., Titarenko, L., Mazurkiewicz, M. and Krzywicki, K. (2021). Improving LUT count of FPGA-based sequential blocks, *Bulletin of the Polish Academy of Sciences: Technical Sciences* **69**(2): 1–12.

Barkalov, A., Titarenko, L. and Mielcarek, K. (2020b). Improving characteristics of LUT-based Mealy FSMs, *International Journal of Applied Mathematics and Computer Science* **30**(4): 745–759, DOI: 10.34768/amcs-2020-0055.

Benini, L. and Micheli, G.d. (2000). System-level power optimization: Techniques and tools, *ACM Transactions on Design Automation of Electronic Systems* **5**(2): 115–192.

BLSG (2005). ABC: A system for sequential synthesis and verification, Berkeley Logic Synthesis and Verification Group, `http://www.eecs.berkeley.edu/~alanmi/abc`.

Bogliolo, A., Benini, L. and De Micheli, G. (1998). Characterization-free behavioral power modeling, *Design, Automation and Test in Europe, Paris, France*, pp. 767–773.

Brooks, D., Tiwari, V. and Martonosi, M. (2000). Wattch: A framework for architectural-level power analysis and optimizations, *ACM SIGARCH Computer Architecture News* **28**(2): 83–94.

Chen, C., Srivastava, A. and Sarrafzadeh, M. (2001). On gate level power optimization using dual-supply voltages, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **9**(5): 616–629.

Cheng, L., Chen, D. and Wong, M.D. (2008). DDBDD: Delay-driven BDD synthesis for FPGAs, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **27**(7): 1203–1213.

Chung, S.J.K. and Brayton, A.M.R. (2009). A power optimization toolbox for logic synthesis and mapping, https://people.eecs.berkeley.edu/~alanmi/publications/2009/iwls09_pwr.pdf.

CBEAL (2004). Collection of digital design benchmarks, https://ddd.fit.cvut.cz/www/prj/Benchmarks/.

Costa, J.C., Monteiro, J.C. and Devadas, S. (1997). Switching activity estimation using limited depth reconvergent path analysis, *Proceedings of the 1997 International Symposium on Low Power Electronics and Design, Monterey, USA*, pp. 184–189.

Curtis, H.A. (1962). *The Design of Switching Circuits*, D. van Nostrand Company, New York.

Dubrova, E. (2004). A polynomial time algorithm for non-disjoint decomposition of multiple-valued functions, *34th International Symposium on Multiple-Valued Logic, Toronto, Canada*, pp. 309–314.

Dubrova, E., Teslenko, M. and Martinelli, A. (2004). On relation between non-disjoint decomposition and multiple-vertex dominators, *2004 IEEE International Symposium on Circuits and Systems, Vancouver, Canada*, Vol. 4, pp. 493–496.

Ferreira, R., Trullemans, A.-M., Costa, J. and Monteiro, J. (2000). Probabilistic bottom-up RTL power estimation, *IEEE 2000 1st International Symposium on Quality Electronic Design, San Jose, USA*, pp. 439–446.

Hasan Babu, H.M. and Sasao, T. (1999). Representations of multiple-output functions using binary decision diagrams for characteristic functions, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **82**(11): 2398–2406.

Jóźwiak, L. and Chojnacki, A. (2003). Effective and efficient FPGA synthesis through general functional decomposition, *Journal of Systems Architecture* **49**(4): 247–265.

Kajstura, K. and Kania, D. (2018). Low power synthesis of finite state machines—State assignment decomposition algorithm, *Journal of Circuits, Systems and Computers* **27**(03): 1850041.

Kim, S. and Kim, J. (2000). Low-power data representation, *Electronics Letters* **36**(11): 1.

Kubica, M., Kajstura, K. and Kania, D. (2018). Logic synthesis of low power FSM dedicated into LUT-based FPGA, *Proceedings of the International Conference of Computational Methods in Sciences and Engineering, Thessaloniki, Greece*, pp. 1–4.

Kubica, M. and Kania, D. (2016). SMTBDD: New form of BDD for logic synthesis, *International Journal of Electronics and Telecommunications* **62**(1): 33–41.

Kubica, M. and Kania, D. (2017a). Area-oriented technology mapping for LUT-based logic blocks, *International Journal of Applied Mathematics and Computer Science* **27**(1): 207–222, DOI: 10.1515/amcs-2017-0015.

Kubica, M. and Kania, D. (2017b). Decomposition of multi-output functions oriented to configurability of logic blocks, *Bulletin of the Polish Academy of Sciences: Technical Sciences* **65**(3): 317–331.

Kubica, M. and Kania, D. (2019). Technology mapping oriented to adaptive logic modules, *Bulletin of the Polish Academy of Sciences: Technical Sciences* **67**(5): 947–956.

Kubica, M., Opara, A. and Kania, D. (2017). Logic synthesis for FPGAs based on cutting of BDD, *Microprocessors and Microsystems* **52**: 173–187, DOI: 10.1016/j.micpro.2017.06.010.

Kubica, M., Opara, A. and Kania, D. (2021a). Logic synthesis strategy oriented to low power optimization, *Applied Sciences* **11**(19): 8797.

Kubica, M., Opara, A. and Kania, D. (2021b). *Technology Mapping for LUT-based FPGA*, Springer, Cham, DOI: 10.1007/978-3-030-60488-2.

Kuc, M., Sułek, W. and Kania, D. (2020). Low power QC-LDPC decoder based on token ring architecture, *Energies* **13**(23): 6310.

Li, X., Chen, L., Yang, F., Yuan, M., Yan, H. and Wan, Y. (2022). HIMAP: A heuristic and iterative logic synthesis approach, *Proceedings of the 59th ACM/IEEE Design Automation Conference, DAC'22, San Francisco, USA*, pp. 415–420, DOI: 10.1145/3489517.3530460.

Lin, Z., Yuan, Z., Zhao, J., Zhang, W., Wang, H. and Tian, Y. (2022). Powergear: Early-stage power estimation in FPGA HLS via heterogeneous edge-centric GNNs, *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe, DATE'22, Antwerp, Belgium*, pp. 1341–1346.

Lindgren, P., Kerttu, M., Thornton, M. and Drechsler, R. (2001). Low power optimization technique for BDD mapped circuits, *Proceedings of the 2001 Asia and South Pacific Design Automation Conference, Yokohama, Japan*, pp. 615–621.

Ling, A., Singh, D.P. and Brown, S.D. (2005). FPGA technology mapping: A study of optimality, *Proceedings of the 42nd Annual Design Automation Conference, DAC'05, New York, USA*, pp. 427–432, DOI: 10.1145/1065579.1065693.

Manzak, A. and Chakrabarti, C. (2002). A low power scheduling scheme with resources operating at multiple voltages, *IEEE Transactions on Very Large Scale Integration Systems* **10**(1): 6–14.

Marakkalage, D.S., Testa, E., Riener, H., Mishchenko, A., Soeken, M. and De Micheli, G. (2020). Three-input gates for logic synthesis, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **40**(10): 2184–2188.

Mehrotra, R. (2013). *Systematic Delay-driven Power Optimisation and Power-driven Delay Optimisation of Combinational Circuits*, PhD thesis, University College Cork, Cork.

Minato, S.-i. (1996). *Binary Decision Diagrams and Applications for VLSI CAD*, Kluwer Academic Publishers, New York.

Opara, A. and Kania, D. (2009). A novel non-disjunctive method for decomposition of CPLDs, *Electronics and Telecommunications Quarterly* **55**(1): 95–111.

Opara, A. and Kubica, M. (2017). Optimization of synthesis process directed at FPGA circuits with the usage of non-disjoint decomposition, *AIP Conference Proceedings*, **1906**: 120004, DOI: 10.1063/1.5012396.

Opara, A. and Kubica, M. (2018). The choice of decomposition path taking non-disjoint decomposition into account, *AIP Conference Proceedings* **2040**, Paper ID: 080010, DOI: 10.1063/1.5079144.

Opara, A., Kubica, M. and Kania, D. (2018). Strategy of logic synthesis using MTBDD dedicated to FPGA, *Integration* **62**: 142–158, DOI: 10.1016/j.vlsi.2018.02.009.

Opara, A., Kubica, M. and Kania, D. (2019). Methods of improving time efficiency of decomposition dedicated at FPGA structures and using BDD in the process of cyber-physical synthesis, *IEEE Access* **7**: 20619–20631, DOI: 10.1109/ACCESS.2019.2898230.

Patalas-Maliszewska, J., Wiśniewski, R., Topczak, M. and Wojnakowski, M. (2022). Design optimization of the Petri net-based production process supported by additive manufacturing technologies, *Bulletin of the Polish Academy of Sciences: Technical Sciences* **70**(2): e140693.

Raghunathan, A., Jha, N.K. and Dey, S. (2012). *High-Level Power Analysis and Optimization*, Springer, New York, USA.

Rawski, M., Łuba, T., Jachna, Z. and Tomaszewicz, P. (2005). The influence of functional decomposition on modern digital design process, *in* M.A. Adamski *et al.* (Eds), *Design of Embedded Control Systems*, Springer US, Boston, pp. 193–204, DOI: 10.1007/0-387-28327-7_17.

Sánchez, F.M., Fungairiño, Y.T. and Alcaide, T.R. (2009). A BDD proposal for probabilistic switching activity estimation, *Proceedings of the 23rd International Conference on Design of Circuits and Integrated Systems (DCIS), Grenoble, France*.

Scholl, C. (2001). *Functional Decomposition with Applications to FPGA Synthesis*, Springer, New York.

Selvaraj, H., Sapiecha, P., Rawski, M. and Łuba, T. (2006). Functional decomposition—The value and implication for both neural networks and digital designing, *International Journal of Computational Intelligence and Applications* **6**(01): 123–138.

Vemuri, N., Kalla, P. and Tessier, R. (2002). BDD-based logic synthesis for LUT-based FPGAs, *ACM Transactions on Design Automation of Electronic Systems* **7**(4): 501–525.

Wisniewski, R. (2021). Design of Petri net-based cyber-physical systems oriented on the implementation in field programmable gate arrays, *Energies* **14**(21): 7054, DOI: 10.3390/en14217054.

Wisniewski, R., Grobelna, I. and Karatkevich, A. (2020). Determinism in cyber-physical systems specified by interpreted Petri nets, *Sensors* **20**(19): 5565.

Wojnakowski, M., Wiśniewski, R., Bazydło, G. and Popławski, M. (2021). Analysis of safeness in a Petri net-based specification of the control part of cyber-physical systems, *International Journal of Applied Mathematics and Computer Science* **31**(4): 647–657, DOI: 10.34768/amcs-2021-0045.

Xilinx (2021). Vivado design suite user guide: Implementation (UG904), `https://docs.xilinx.com/r/en-US/ug904-vivado-implementation`.

Adam Opara received his MSc and PhD degrees from the Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Poland, in 2002 and 2009, respectively, where he is currently an assistant professor with the Department of Graphics, Computer Vision and Digital Systems. His research interests include programmable digital devices, microprocessors, logic synthesis, and technology mapping.

Marcin Kubica received his MSc and PhD degrees from the Silesian University of Technology, Gliwice, Poland, in 2010 and 2014, respectively, where he has been an assistant professor with the Department of Digital Systems. His main research interests include programmable devices and systems, and logic synthesis.

Received: 27 June 2022
Revised: 24 October 2022
Re-revised: 30 November 2022
Accepted: 12 January 2023