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 Raffa, 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
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 good time efficiency of the implementation of specific approach allows easy prototyping of circuits and ensures FPGA (field programmable gate array) devices. This adjusted clock frequency are locally located.

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 BLG (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 PMTBBDD; 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 \( \text{numb}(\omega_f \omega_g) \) 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 \( \text{numb}(\omega_f \omega_g) \) 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
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 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
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(b). If any of the previously created clusters. Then, the number of variables on which the given function is dependent is determined \((\text{num}\text{b}_d f_{\text{w} \text{h} \text{e} \text{r} f})\) for functions not connected to any cluster \( f_0 \) and the given function \( f_1 \). 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 \( \text{numb}_d f_{\text{w} \text{h} \text{e} \text{r} f} \) 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 \( \text{numb}_d f_{\text{w} \text{h} \text{e} \text{r} f} \leq 0 \) for the case where it is not necessary to decompose the given function \((\text{numb}_d f_{\text{w} \text{h} \text{e} \text{r} f} \leq k)\) or if \( \text{numb}_d f_{\text{w} \text{h} \text{e} \text{r} f} \leq \text{numb}_d f_{\text{w} \text{h} \text{e} \text{r} f, \text{com}} \). 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 \( X_0 \) 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 \( \text{bdd}\_\text{and}(\cdot) \) and \( \text{bdd}\_\text{compose}(\cdot) \), 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(a) (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
cases are marked in grey. As a result of merging both diagrams (bdd_or), a PMTBD representation of the functions $f_0$ and $f_1$, 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 PMTBD 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 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 is 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 PMTBD diagram to include additional information necessary for its reduction.

5. SWMTBDDs and SWPMTBDDs

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

$$\text{P}_{\text{dyn}} = \text{P}_{\text{stat}} + \text{P}_{\text{dyn}}.$$  

The static power $\text{P}_{\text{stat}}$ 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 $\text{P}_{\text{dyn}}$. The value of dynamic power can be determined from

$$\text{P}_{\text{dyn}} = \frac{1}{2} V_{\text{dd}}^2 f \sum_{i=1}^{n} C_i \text{SW}_i.$$  

Analyzing (2), the dynamic power $\text{P}_{\text{dyn}}$ depends on a number of parameters, such as the supply voltage of the circuit $V_{\text{dd}}$ 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 ($\text{SW}_i$). 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

$$SW = 2P(x)(1 - P(x)),$$

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. 5b.

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 $x$, the following formula is applied:

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

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

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

(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:

$$ITE(f, g, h) = ITE(x_i),$$

$$ITE(f_{x_i=1}, g_{x_i=1}, h_{x_i=1}),$$

$$ITE(f_{x_i=0}, g_{x_i=0}, h_{x_i=0}).$$

(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.
The standard operator ITE is used for a logic sum operation. Making some modifications to the ITE algorithm, the probability can be retained. The second diagram proposed by the authors is the 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):

$$f_0 f'_0 + f_1 f'_1.$$  

(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

**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: $R.P[i] = P(x) \times PosFactor.P[i] + (1 - P(x)) \times NegFactor.P[i]$;
15: end for
16: end if
17: InsertIntoResultTable($(I,T,E), R$);
18: return $R$;
19: end if
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 $\text{num}_g - \text{sum}_\text{card}_X_b$ has the greatest weight and $\text{SW}_{\text{cost}}\text{sum}$ the least. Based on experiments, the $\text{SW}_{\text{cost}}\text{sum}$ is always smaller than $2^7$; therefore, the shift is seven bits. In addition, $\text{num}_{\text{nndisj}}$ is always less than $2^5$; therefore, the offset is five bits. The last step is 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 reorganize the order of variables. The next step is the implementation of the decomposition algorithm aimed at minimizing dynamic power, which is shown in Fig. 2.
The essence of this algorithm is the successive change in the order of the variables performed using the \( \text{swap}(x_i, x_j, F) \) 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 (\( \text{Cost}\text{temp} \)) is estimated, which is then compared with the best-known solution (\( \text{Cost} \)). 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 “\( \text{with} \)non – dis.dec.” and without “\( \text{without} \)non – dis.dec.” Table 1 lists the number of LUT blocks “\( \text{LUT}_5 \)”, the number of shared bound functions \( g \) “\( \text{Shared} \)”, the number of logic levels “\( \text{Levels} \)”, and the switching activity “\( \text{SW} \)”. In addition, information about the synthesis time “\( \text{time} \)” (expressed in seconds) and the value of the fanout “\( \text{fanout} \)” are also provided. Additionally, in the section “\( \text{with} \)non – dis.dec.” the number of non-disjoint decompositions “\( \text{Ndisj} \)” is given.

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
Technology mapping of multi-output functions leading to the reduction of

<table>
<thead>
<tr>
<th>Function</th>
<th>SW</th>
<th>LUT_5 Shared Levels time fanout SW</th>
<th>LUT_5 N disj Shared Levels time fanout SW</th>
<th>Powerdek/MF</th>
<th>DekBDD PowerdekBDD_MF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benchm.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>apex7</td>
<td>49</td>
<td>37</td>
<td>130</td>
<td>103</td>
<td>27</td>
</tr>
<tr>
<td>clip</td>
<td>9</td>
<td>5</td>
<td>18</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>cm85a</td>
<td>11</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>count</td>
<td>35</td>
<td>16</td>
<td>44</td>
<td>35</td>
<td>12</td>
</tr>
<tr>
<td>example2</td>
<td>85</td>
<td>66</td>
<td>124</td>
<td>88</td>
<td>22</td>
</tr>
<tr>
<td>f51m</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>inc</td>
<td>7</td>
<td>9</td>
<td>20</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>misex1</td>
<td>8</td>
<td>7</td>
<td>15</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>misex2</td>
<td>25</td>
<td>18</td>
<td>33</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>misex3c</td>
<td>14</td>
<td>14</td>
<td>134</td>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td>pdc</td>
<td>16</td>
<td>40</td>
<td>117</td>
<td>42</td>
<td>7</td>
</tr>
<tr>
<td>rd53</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rd73</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>sao2</td>
<td>10</td>
<td>4</td>
<td>33</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>sct</td>
<td>19</td>
<td>15</td>
<td>20</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>spla</td>
<td>16</td>
<td>46</td>
<td>372</td>
<td>133</td>
<td>32</td>
</tr>
<tr>
<td>squar5</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>vg2</td>
<td>25</td>
<td>8</td>
<td>72</td>
<td>46</td>
<td>5</td>
</tr>
<tr>
<td>x4</td>
<td>94</td>
<td>71</td>
<td>109</td>
<td>41</td>
<td>18</td>
</tr>
</tbody>
</table>

Sum: 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

Geometric mean: 3.45 0.92 0.04 2.87 1.93 3.93 0.93 0.04 3.08 1.72 3.57 0.66 0.05 2.85 1.35 3.07 0.68 0.06 3.08 1.27

Table 1. Comparison of the DekBDD and Powerdek-MF Systems.
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.

<table>
<thead>
<tr>
<th>Bench.</th>
<th>Inputs</th>
<th>Outputs</th>
<th>PowerdekBDD_MF (with non-disj. dec.)</th>
<th>ABC Baseline</th>
<th>ABC PowerMap</th>
<th>ABC PowerDC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LUT_5 Levels Power</td>
<td>LUT_5 Levels Power</td>
<td>LUT_5 Levels Power</td>
<td></td>
</tr>
<tr>
<td>5xp1</td>
<td>7</td>
<td>10</td>
<td>13  2  30.80</td>
<td>23</td>
<td>3</td>
<td>7,86</td>
</tr>
<tr>
<td>5sym</td>
<td>9</td>
<td>3</td>
<td>6   9  24.58</td>
<td>67</td>
<td>5</td>
<td>131.85</td>
</tr>
<tr>
<td>b24</td>
<td>16</td>
<td>10</td>
<td>8   7  70.75</td>
<td>193</td>
<td>6</td>
<td>207.75</td>
</tr>
<tr>
<td>b54</td>
<td>14</td>
<td>8</td>
<td>4   3  925.03</td>
<td>177</td>
<td>6</td>
<td>629.28</td>
</tr>
<tr>
<td>apex7</td>
<td>49</td>
<td>37</td>
<td>9   8  170.08</td>
<td>104</td>
<td>4</td>
<td>189.59</td>
</tr>
<tr>
<td>b12</td>
<td>15</td>
<td>9</td>
<td>18  2  40.27</td>
<td>18</td>
<td>2</td>
<td>39.04</td>
</tr>
<tr>
<td>b9</td>
<td>16</td>
<td>5</td>
<td>45  5  86.12</td>
<td>23</td>
<td>3</td>
<td>47.57</td>
</tr>
<tr>
<td>bw</td>
<td>5</td>
<td>28</td>
<td>28  1  77.17</td>
<td>28</td>
<td>1</td>
<td>77.82</td>
</tr>
<tr>
<td>c8</td>
<td>28</td>
<td>18</td>
<td>29  4  65.51</td>
<td>32</td>
<td>3</td>
<td>65.91</td>
</tr>
<tr>
<td>cht</td>
<td>47</td>
<td>36</td>
<td>37  2  92.59</td>
<td>37</td>
<td>2</td>
<td>92.97</td>
</tr>
<tr>
<td>cip</td>
<td>6</td>
<td>18</td>
<td>4   3  63.14</td>
<td>21</td>
<td>3</td>
<td>63.43</td>
</tr>
<tr>
<td>cm16a</td>
<td>14</td>
<td>5</td>
<td>11  3  21.74</td>
<td>11</td>
<td>3</td>
<td>21.13</td>
</tr>
<tr>
<td>cm16b</td>
<td>16</td>
<td>5</td>
<td>9   3  19.49</td>
<td>9</td>
<td>2</td>
<td>19.71</td>
</tr>
<tr>
<td>cm58a</td>
<td>7</td>
<td>3</td>
<td>8   3  16.63</td>
<td>11</td>
<td>2</td>
<td>16.83</td>
</tr>
<tr>
<td>count</td>
<td>35</td>
<td>42</td>
<td>6   9  80.36</td>
<td>49</td>
<td>3</td>
<td>83.07</td>
</tr>
<tr>
<td>duke2</td>
<td>22</td>
<td>29</td>
<td>25  10 443.89</td>
<td>178</td>
<td>4</td>
<td>294.81</td>
</tr>
<tr>
<td>c64</td>
<td>65</td>
<td>64</td>
<td>10  8  122.57</td>
<td>191</td>
<td>4</td>
<td>241.75</td>
</tr>
<tr>
<td>example2</td>
<td>85</td>
<td>66</td>
<td>123 6 692.28</td>
<td>377</td>
<td>6</td>
<td>669.28</td>
</tr>
<tr>
<td>f51m</td>
<td>8</td>
<td>8</td>
<td>10  3  22.73</td>
<td>26</td>
<td>3</td>
<td>56.56</td>
</tr>
<tr>
<td>imc</td>
<td>7</td>
<td>9</td>
<td>20  3  44.56</td>
<td>25</td>
<td>3</td>
<td>53.26</td>
</tr>
<tr>
<td>misex1</td>
<td>25</td>
<td>18</td>
<td>32  4  63.97</td>
<td>35</td>
<td>3</td>
<td>64.71</td>
</tr>
<tr>
<td>misex2</td>
<td>14</td>
<td>29</td>
<td>11  5  584.78</td>
<td>382</td>
<td>6</td>
<td>668.89</td>
</tr>
<tr>
<td>misexc</td>
<td>14</td>
<td>125</td>
<td>10  17 231.78</td>
<td>194</td>
<td>5</td>
<td>358.97</td>
</tr>
<tr>
<td>pdc</td>
<td>16</td>
<td>120</td>
<td>28  4  241.88</td>
<td>596</td>
<td>6</td>
<td>1082.39</td>
</tr>
<tr>
<td>rds3</td>
<td>5</td>
<td>3</td>
<td>1  3  6.60</td>
<td>17</td>
<td>2</td>
<td>34.89</td>
</tr>
<tr>
<td>rd73</td>
<td>7</td>
<td>6</td>
<td>2   3  14.34</td>
<td>16</td>
<td>3</td>
<td>38.91</td>
</tr>
<tr>
<td>rd84</td>
<td>8</td>
<td>9</td>
<td>3   1  21.05</td>
<td>63</td>
<td>4</td>
<td>122.25</td>
</tr>
<tr>
<td>sza2</td>
<td>10</td>
<td>4</td>
<td>33  6  56.56</td>
<td>36</td>
<td>3</td>
<td>72.25</td>
</tr>
<tr>
<td>spu</td>
<td>11</td>
<td>15</td>
<td>17  7  40.72</td>
<td>10</td>
<td>1</td>
<td>44.72</td>
</tr>
<tr>
<td>spla</td>
<td>16</td>
<td>128</td>
<td>11  8  434.34</td>
<td>363</td>
<td>5</td>
<td>624.5</td>
</tr>
<tr>
<td>squar5</td>
<td>5</td>
<td>8</td>
<td>1   3  20.18</td>
<td>8</td>
<td>1</td>
<td>20.18</td>
</tr>
<tr>
<td>t481</td>
<td>16</td>
<td>5</td>
<td>4   1  10.23</td>
<td>20</td>
<td>4</td>
<td>31.05</td>
</tr>
<tr>
<td>table3</td>
<td>14</td>
<td>14</td>
<td>813 15 1467.82</td>
<td>568</td>
<td>6</td>
<td>894.72</td>
</tr>
<tr>
<td>term1</td>
<td>34</td>
<td>10</td>
<td>30  5  52.78</td>
<td>39</td>
<td>4</td>
<td>77.98</td>
</tr>
<tr>
<td>ttt2</td>
<td>24</td>
<td>21</td>
<td>40  3  82.52</td>
<td>47</td>
<td>3</td>
<td>88.88</td>
</tr>
<tr>
<td>vg3</td>
<td>25</td>
<td>8</td>
<td>65  7  134.68</td>
<td>40</td>
<td>4</td>
<td>166.06</td>
</tr>
<tr>
<td>x2</td>
<td>10</td>
<td>7</td>
<td>15  3  29.95</td>
<td>13</td>
<td>2</td>
<td>38.36</td>
</tr>
<tr>
<td>x4</td>
<td>94</td>
<td>71</td>
<td>113 5 234.38</td>
<td>115</td>
<td>3</td>
<td>208.65</td>
</tr>
<tr>
<td>Sum:</td>
<td>3455</td>
<td>213</td>
<td>6754.03</td>
<td>3996</td>
<td>137</td>
<td>7082.78</td>
</tr>
<tr>
<td>Geometric mean:</td>
<td>30.57</td>
<td>4.06</td>
<td>64.98</td>
<td>39.80</td>
<td>2.98</td>
<td>78.85</td>
</tr>
</tbody>
</table>

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
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. 12b). 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
synthesized in Vivado. Results were obtained regarding the number of LUT blocks used as “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, ABC PowerMap, ABC PowerDC, and Vivado tools.
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 “with non − dis.dec.” and without non-disjoint decomposition “without non − 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


Technology mapping of multi-output functions leading to the reduction...


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.