IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2018, accepted January 11, 2019, date of publication January 25, 2019, date of current version February 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2895206

A Technology Mapping of FSMs Based on a Graph of Excitations and Outputs

MARCIN KUBICA, DARIUSZ KANIA, AND JÓZEF KULISZ
Institute of Electronics, Silesian University of Technology, 44-100 Gliwice, Poland
Corresponding author: Marcin Kubica (marcin.kubica@polsl.pl)

This work was supported in part by the Polish Ministry of Science and Higher Education.

ABSTRACT A logic synthesis for finite-state machines (FSMs) aimed at programmable array logic (PAL)-based complex programmable logic devices is proposed here. This approach consists of the simultaneous synthesis of a transition function and an output function. The main contribution is the novel multilevel optimization of an FSM. In this process, a new form of graph is used, i.e., a graph of excitations and outputs. This is a generalization of the graph of outputs that has previously been used in the process of technology mapping of multi-output functions in PAL-based programmable structures. The main idea, the theoretical background, and a precise algorithm are illustrated by means of simple examples. The proposed algorithm was compared with other approaches by synthesizing the FSM benchmarks and mapping the solutions to *k*-term PAL-based logic blocks. The obtained results are compared on the basis of the area (number of logic blocks) and speed (number of logic levels). The proposed approach is especially effective for larger FSMs.

INDEX TERMS CPLD, FSM, multi-level optimization, technology mapping.

I. INTRODUCTION

The structure of the cyber-physical systems presented in [1] is layered. The lowest layer, the 'physical system', may use a hardware implementation and a hardware description language. Similarly, advanced sensors and complex communication tasks may be involved in the area of the Internet of Things (IoT) [2]. One of the possibilities for carrying out these implementations in cyber-physical systems is the use of complex programmable logic device (CPLD) circuits. Thus, it may be said that a crucial aspect of the effective synthesis of cyber-physical systems is an efficient logic synthesis dedicated to CPLD circuits, and this is the main topic of this paper.

The minimization of logic resources, which is needed to implement projects in CPLD, is especially vital in the process of logic synthesis. It can lead to a reduction in the costs for an implemented project in a 'physical system', and can reduce the costs of cyber-physical systems and limit power consumption, which is especially important in IoT applications. Since most implementations include complex sequential circuits, it is necessary to develop finite state machine (FSM) synthesis methods to minimize the usage of logic blocks in CPLD, and this is the essence of this paper.

A programmable array logic (PAL)-based logic block, as presented in Fig. 1, constitutes the kernel of a classical CPLD. It is generally possible to configure the block for a combinatorial or registered operation (i.e. to a form with or without a D flip-flop), and this is also illustrated in Fig. 1. Logic synthesis for an FSM starts with a state assignment, and a two-level minimization is then executed. Following this, minimization of each single-output function is carried out separately [3] in most approaches, and technology mapping is then started. Mapping of the minimized functions in PAL-based blocks containing a predefined number of product terms with and without the D F-Fs is applied.

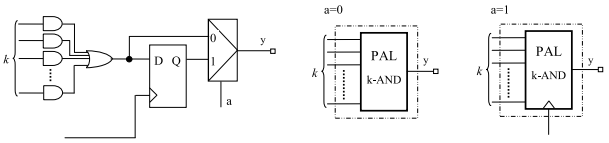

FIGURE 1. Structure of a k-term PAL-based logic block, and symbols representing the block configured for combinatorial or registered operation.

Internal state assignment is a vital stage of FSM synthesis, and is usually directly associated with the result that we expect to obtain. The goal of optimization may be the chip area required for the FSM [4], [5], the speed of operation of the automaton [5], [6], or minimization of the power consumption [7]–[9]. Issues concerning the minimization of power consumption have recently become particularly important. Initially, methods were proposed that were based on

the minimization of transitions of the state variables; this was achieved by grouping the states between which frequent transitions occur, by assigning to them codes with a minimum code distance (the minimum weighted Hamming distance method) [10]–[12]. The problem of state assignment can be considered in terms of integer linear programming [10], but heuristic methods are generally used [13], [14]. Genetic algorithms are sometimes applied in the process of coding [15], [16]. The search for more efficient implementations of automatons is sometimes carried out also in the form of a globally asynchronous locally synchronous (GALS) [17] structure, or solutions exploiting clock gating [18], [19].

In the case of CPLDs, the most important aspect, apart from the process of state assignment, is the ability to efficiently use the product terms contained in the PAL-based logic blocks. Minimization of the number of implicants constitutes an integral part of coding algorithms, as it can influence the minimization of the area, the number of logic levels and even the minimization of power consumption.

Previous papers presented by the current authors propose a number of optimization methods dedicated to combinational circuits, such as those using graphs of outputs [20], [21], methods of technology mapping exploiting tri-state output buffers [22], and various decomposition strategies [23], [24]. Binary decision diagrams are sometimes used in the synthesis process [24]–[27], and these methods are directly associated with the way the state assignment is carried out [3], [22], [28]. However, in each case, the algorithms are based on a separate synthesis of the transition function and the output function. A two-level minimization of the transition and the output functions is typically performed. In most well-known approaches, each single-output function is minimized separately [3]. If the number of implicants p, representing a function after minimization, is greater than the number of product terms k in a logic block (Fig. 1), a greater number of logic blocks needs to be used. Multilevel synthesis dedicated to CPLD has sometimes been reported [5], but the majority of multi-level logic synthesis techniques are dedicated to programmable logic array (PLA)-based devices or look-up table (LUT)-based field programmable gate arrays (FPGAs) [29]–[32].

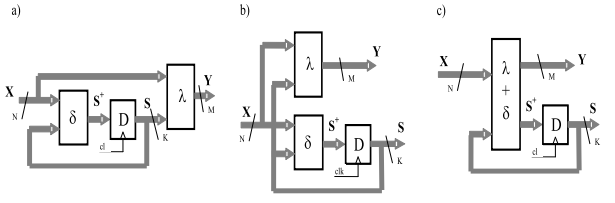

FIGURE 2. Block diagrams representing the internal structure of an FSM.

Multilevel optimization has become very important, especially when a function is implemented in CPLD structures. A block diagram of an FSM is presented in Fig. 2a. Its form suggests that it is necessary to implement two separate multi-output functions, describing the transition block δ and the output block λ, within a programmable structure. A simple modification of the structure, from the form presented in Fig. 2b to that illustrated in Fig. 2c, shows that it is possible to describe the two combinational blocks using a single multi-output function. The question is now whether it efficient to simultaneously optimize the whole combinational block, i.e. the transition block δ and an output block λ together. A simultaneous synthesis of the transition block and the output block gives better optimization conditions than the separate synthesis of each. Fortunately, D-type flip-flops make the synthesis process easier, as these are based on the optimization of a multi-output function in which some signals form the input vector, and others the transition vector that describes the signals connected to the inputs of the flip-flops.

The aim of this paper is to present a method of FSM synthesis dedicated to PAL-based CPLD, the essence of which consists of a novel multilevel optimization of FSM involving the simultaneous technology mapping of the transition and the output blocks. In this process of technology mapping, a new form of a circuit description is used, referred to here as the graph of excitations and outputs.

II. THEORETICAL BACKGROUND

A sequential automaton is described by determining five elements $\{X, Y, S, \delta, \lambda\}$, where X is the set of input elements, Y is the set of output elements, S is the set of the internal states of the automaton, δ is the transition function and λ is the output function [34]. Inputs are characterized as N-bit input vectors, where $N \geq \lceil \lg_2(\mathrm{card}(X)) \rceil$, and card(X) means the cardinality of set X. Similarly, outputs are defined as M-bit output vectors, where $M \geq \lceil \lg_2(\mathrm{card}(Y)) \rceil$. In the process of state assignment, a K-bit state vector, where $K \geq \lceil \lg_2(\mathrm{card}(S)) \rceil$, is assigned to each symbolic state of the automaton [34].

In accordance with the notation presented above, a transition function characterized as the mapping δ: $X \times S \rightarrow S^+$ should be associated with the function δ: $B^{N+K} \rightarrow B^K$, where the output is $B = \{0,1\}$. The form of the output function depends on the type of the automaton. In the case of Moore's automaton, the form of the function is characterized as the relation between outputs and the set of internal states, i.e. λ: $S \rightarrow Y$. In the case of Mealy's automaton, the input signals λ: $X \times S \rightarrow Y$ also determine the states of the outputs. As in the case of the transition function, the output function may be associated with the mapping λ: $B^K \rightarrow B^M$ for Moore's automaton, and λ: $B^{N+K} \rightarrow B^M$ for Mealy's automaton.

A simple and convenient form for representing an FSM is a textual description in the KISS format [33]. This consists of symbolic implicants that include the input part, a symbolic description of the present state, a symbolic description of the next state, and the output states. In general, the form of the KISS description follows Mealy's scheme for an automaton (Fig. 2a).

Example 1: Let us consider an exemplary sequential automaton described in the KISS format. The input signals

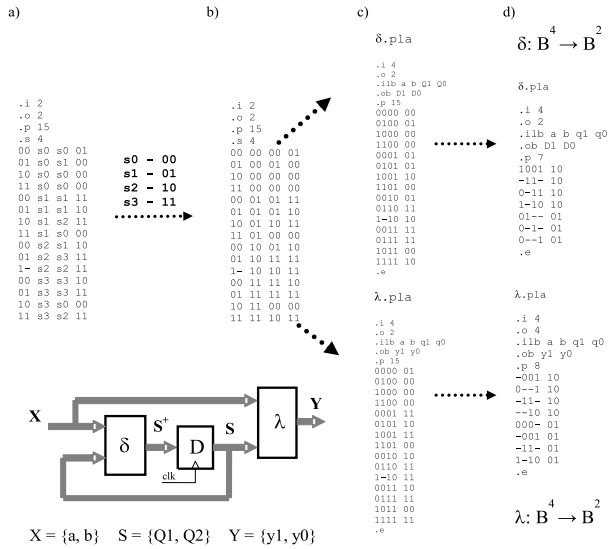

FIGURE 3. The logic synthesis of an FSM without multilevel optimization.

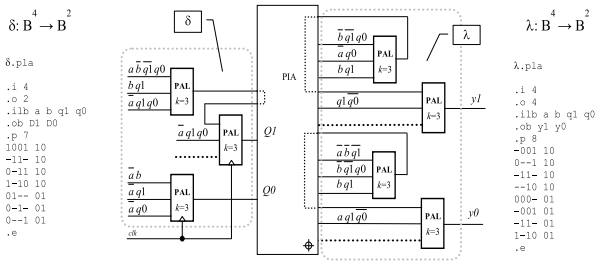

FIGURE 4. The technology mapping of an FSM implemented in a PAL-based device.

are denoted by the letters a and b, and the output signals by the symbols $y1$ and $y0$ respectively. The automaton contains four states, and its operation is described using 12 symbolic multi-output implicants (Fig. 3a). In the process of state assignment, the symbolically described states S0, S1, S2, S3 are associated with two-bit binary vectors forming coding states 00, 01, 10, and 11. This leads to the description of the automaton after the state assignment, as illustrated in Fig. 3b. In the next stage, the synthesis of the transition and the output blocks is carried out. Descriptions of the transition function δ: $B^4 \rightarrow B^2$, and the output function λ: $B^4 \rightarrow B^2$ in .pla format [34] are presented in Fig. 3c. Descriptions of separate functions may be obtained by appropriate partitioning of the output part of the implicants determining the transitions in the analyzed FSM.

Let us assume that an implementation of the FSM using PAL-based logic blocks containing three product terms is sought. The technology mapping of the transition and output blocks onto PAL-based structures is usually based on the minimization of transition functions δ_i : $B^4 \rightarrow B$ $(i = 1, 2)$ and output functions λ_i: $B^4 \rightarrow B$ $(i = 1, 2)$. All single-output functions are minimized separately. The results of this minimization are presented in Fig. 3d.

The implementation of FSM by means of a network of three-term PAL-based logic blocks is presented in Fig. 4.

CPLD circuits available on the market contain hardware mechanisms that facilitate mapping onto device resource functions that require more product terms than the k terms available in a PAL-based block. In particular, these include parallel expanders and logic allocators. These structures offer the possibility of virtually increasing the number of terms available in a PAL-based block by "borrowing" them from neighboring blocks. In this way, it is possible to increase the number of terms in a cell at the expense of limiting the number of terms available in neighboring blocks. Often, using a greater number of terms means that it is impossible to use neighboring blocks at all. An additional cost of using these expanders is a slight increase in the propagation delay.

In the proposed method, the presence of the expanders could be modeled by using PAL-based blocks that have different numbers of terms. However, for reasons of simplicity this option will be neglected in this paper.

The implementation of combinational blocks in PAL-based CPLD structures, as illustrated in the example, is directly associated with the process of minimization of separate transition and output functions, which is usually carried out using the Espresso-Dso algorithm. In this minimization process, a search for the minimal covering of K transition functions δ_i: $B^{N+K} \rightarrow B$ $(i = 1, \ldots, K)$, and M output functions λ_i: $B^{N+K} \rightarrow B$ $(i = 1, \ldots, M)$ is carried out. After minimization, technology mapping of the minimized forms of separate functions on the structure consisting of PAL-based logic blocks is performed. Let us assume that the PAL-based logic blocks contain k terms. In this situation, the numbers of blocks required to implement the transition and output functions can be calculated using Equations (1) and (2) respectively:

$$\sigma_\delta = \sum_{i=1}^{K} \left(H\left(\left\lceil \frac{\Delta_{\delta i} - k}{k-1} \right\rceil \right) \left\lceil \frac{\Delta_{\delta i} - k}{k-1} \right\rceil + 1 \right) \quad (1)$$

$$\sigma_\lambda = \sum_{i=1}^{M} \left(H\left(\left\lceil \frac{\Delta_{\lambda i} - k}{k-1} \right\rceil \right) \left\lceil \frac{\Delta_{\lambda i} - k}{k-1} \right\rceil + 1 \right) \quad (2)$$

where $\Delta_{\delta i}$, $\Delta_{\lambda i}$ denote the numbers of implicants describing individual single-output transition and output functions, and σ_δ, σ_λ represent the numbers of k-term PAL-based blocks required to implement the transition and the output blocks, respectively. The H symbol denotes the unit step function, and is defined as follows:

$$H(x) = \begin{cases} 0 & for\ x\ < 0 \\ 1 & for\ x\ \geq\ 0 \end{cases}$$

The aim of this paper is to present a technology-dependent optimization of FSMs. The main contribution is a novel multilevel optimization of FSM oriented towards PAL-based CPLDs.

The essence of the proposed method consists of searching for shared implicants of the transition and output functions.

III. FSM DESCRIPTION BY MEANS OF THE GRAPH OF EXCITATIONS AND OUTPUTS

After the state assignment is carried out, an FSM can be considered as a circuit described by two multi-output functions $\delta : B^{N+K} \rightarrow B^K$ and $\lambda : B^{N+K} \rightarrow B^M$, $B = \{0,1\}$, where N is the number of FSM inputs, K is the length of the state vector, and M is the number of FSM outputs. The first function describes the transition block, and the second the output block (Fig. 2a, 2b). The combinational part of an FSM can be described using a single multi-output function that simultaneously describes the two blocks. Let $\delta + \lambda : B^{N+K} \rightarrow B^{K+M}$ be the function describing the combinational block of the structure shown in Fig. 2c.

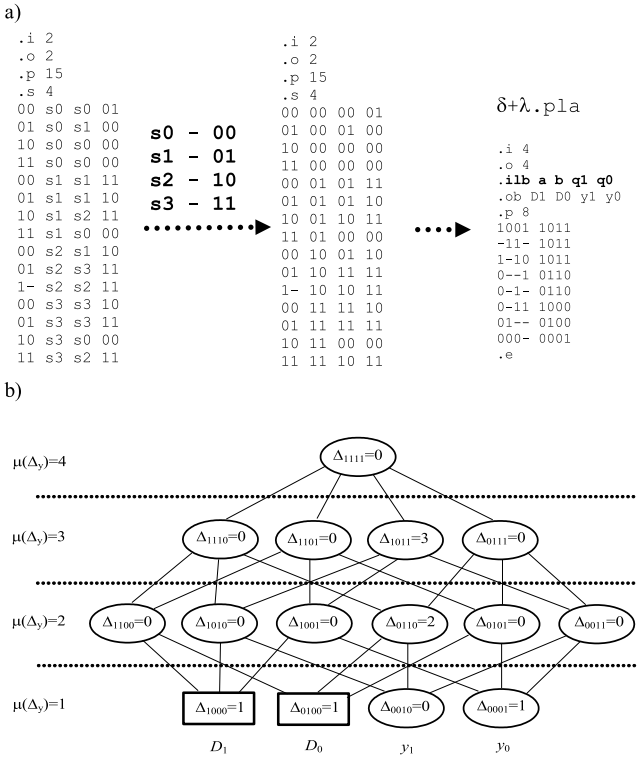

FIGURE 5. Representation of the minimized form of the function f: $B^4 \rightarrow B^4$ by means of the primary graph of excitations and outputs.

This function $\delta + \lambda : B^{N+K} \rightarrow B^{K+M}$ can be represented by a set of multi-output implicants [34]. Let y be the output part of a multi-output implicant consisting of zero or one, and let discriminant Δ_y be the number of the same y output vectors. The range of discriminant Δ_y, denoted as $\mu(\Delta_y)$ is the number of ones contained in y. Let G<Y,U> be the primary graph of excitations and outputs (Fig. 5). Graph nodes are associated with the corresponding discriminants Δ_y, while the edges connect the nodes Δ_{ys}, Δ_{yr}, for which the code distance between the y_s, y_r is one, and $\mu(\Delta_{ys})+1 = \mu(\Delta_{yr})$. The graph of excitations and outputs contains two types of nodes, which are referred to here as combinational nodes (denoted by ellipses on the graph) and sequential nodes (rectangles). Sequential nodes lie on paths that are associated with the excitation functions of particular flip-flops.

Example 2: Let us consider the automaton from Example 1 again. After coding the internal states, and after minimization of the multi-output function $\delta + \lambda$: $B^{N+K} \rightarrow B^{K+M}$, where $N = 2$, $K = 2$, and $M = 2$, we obtain the minimized form of the multi-output function $\delta + \lambda$: $B^4 \rightarrow B^4$, which contains eight implicants. In the set of all implicants, there are two larger groups of implicants: the first contains three implicants with an output part equal to 1011 (1001 1011; -11- 1011; 1-10 1011), and the second contains two implicants with an output part equal to 0110 (0–1 0110; 0-1- 0110). These functions can be represented by the graph of excitations and outputs presented in Fig. 5b.

The graph shown in Fig. 5 contains a number of nodes for which the value of the discriminant $\Delta_y = 0$. Such nodes can be deleted from the primary graph. In this way, a reduced graph of excitations and outputs is created (Fig. 6).

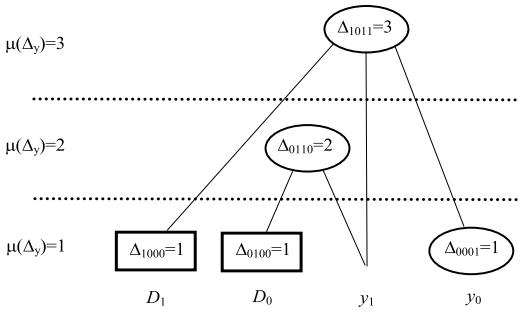

FIGURE 6. The reduced graph of excitations and outputs, representing the minimized form of the example function.

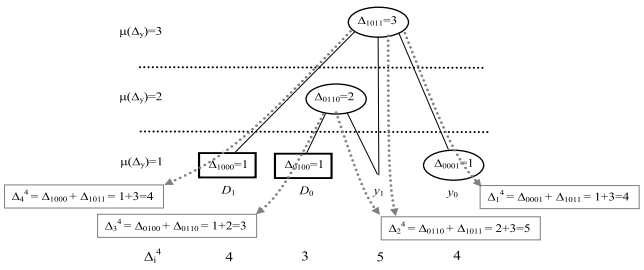

FIGURE 7. The method of determining the values of the discriminants Δ_i^{K+M} ($i = 1, \ldots, 4$; K = 2, M = 2).

By using the values of the discriminants at the nodes of the graph of excitations and outputs, it is possible to determine the number of k-term PAL-based blocks required to implement the FSM. A discriminant Δ_i^{K+M} $(i = 1, 2, \ldots, K+M)$ can be assigned to each of the $\delta_i : B^{N+K} \rightarrow B(i = 1, \ldots, K)$ and λ_i: $B^{N+K} \rightarrow B(i = 1, , M)$ functions that create the multi-output function $\delta + \lambda : B^{N+K} \rightarrow B^{K+M}$. The value of the discriminant Δ_i^{K+M} is equal to the sum of the values of the discriminants Δ_y for which a value of one is present at the same position i in the output part y of the implicants. The method of determining the values of discriminants Δ_i^{K+M} is based on an analysis of the appropriate paths in the graph of excitations and outputs. The values of the discriminants Δ_i^{K+M} and the corresponding paths in the graph from Fig. 6 are presented in Fig. 7.

The number of logic blocks for implementation of the function $\delta + \lambda : \mathrm{B}^{N+K} \rightarrow \mathrm{B}^{K+M}$ is equal to $\sigma_{\delta+\lambda}$ (3).

$$\sigma_{\delta+\lambda} = \sum_{i=1}^{M+K} \left(H\left(\left\lceil \frac{\Delta_i^{M+K} - k}{k-1} \right\rceil \right) \left\lceil \frac{\Delta_i^{M+K} - k}{k-1} \right\rceil + 1 \right) \tag{3}$$

It should be noted that Equation (3) describes the worst case, i.e. the number of blocks required to implement the FSM assuming that each function is built separately, and that co-sharing of blocks does not occur. During further optimization, blocks that can be shared between functions are identified. The essence of the proposed method consists of searching for the solution in which the number of blocks required to implement the FSM is less than the value of the sum $\sigma_\delta + \sigma_\lambda$, where σ_δ and σ_λ are described by Eqs. (1) and (2). This will be described in the next section.

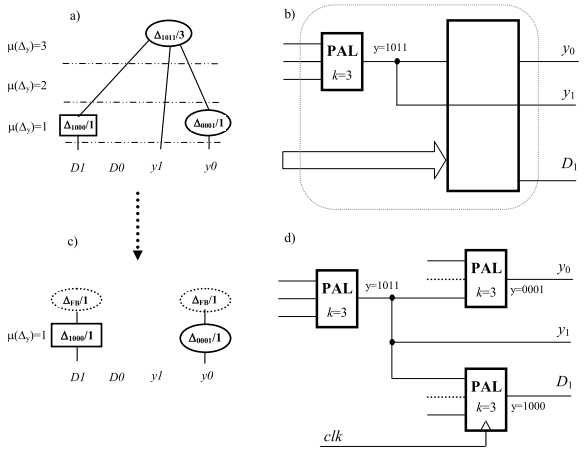

FIGURE 8. An example of the proposed technology mapping: a) an example of a graph of excitations and outputs; b) implementation of implicants associated with the third range node; c) the reduced graph of excitations and outputs; d) the implementation of implicants corresponding to first range nodes and feedbacks.

IV. A TECHNOLOGY MAPPING ALGORITHM BASED ON THE GRAPH OF EXCITATIONS AND OUTPUTS

By analyzing the structure of the graph of excitations and outputs, multi-level optimization is possible.

Example 3: Let us consider the fragment of the graph of excitations and outputs shown in Fig. 8a. Assuming that we use PAL-based logic blocks containing three product terms, the implicants associated with the third range node can be configured in one block (Fig. 8b). Analysis of the graph is an iterative process. If a node of the graph is chosen in step i, the graph needs to be modified. The node Δ_{1011} is eliminated from the graph, and new, additional nodes Δ_{FB} (feedback) are added in the branches, through which the node Δ_{1011} is connected to the nodes in the lower ranges (Fig. 8c). These nodes, which represent the feedback Δ_{FB}, appear for all the functions that have not yet been synthesized.

In the next stage, implementation of the nodes of the first range $\Delta_{1000}$and Δ_{0001} is carried out, leading to the mapping illustrated in Fig. 8d.

To summarize Example 3, in general, the modification of the graph of excitations and outputs that results from the implementation in step i of the implicants associated with node ${}^{i}\Delta_{\mathrm{y}}$ includes the following steps:

- deleting the node ${}^{i}\Delta_{\mathrm{y}}$ from the graph;
- deleting all the edges connecting the node ${}^{i}\Delta_{\mathrm{y}}$ with nodes in lower ranges; and
- creating nodes representing feedbacks, denoted by the symbol Δ_{FB}, and replacing the edges connecting the node ${}^{i}\Delta_{\mathrm{y}}$ with nodes of lower ranges (which were deleted in the previous step) by edges connecting the Δ_{FB} nodes with the nodes of lower ranges. If for a given function an edge connected to a feedback node is to be deleted, a new feedback node is not added to the graph, and instead, the modification consists of incrementing the value of the feedback discriminant, i.e. $\Delta_{\mathrm{FB}} := \Delta_{\mathrm{FB}} + 1$.

As a consequence of the reduction of the graph, the values of the discriminants $\Delta_i^{K+M} (i = 0, \ldots, K+M)$ are also reduced. Figure 9 presents an example modification of the graph that results from deletion of the node Δ_{1011} and its influence on the values of the discriminants Δ_i^{K+M}.

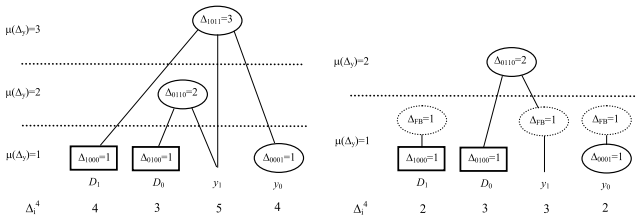

FIGURE 9. The influence of reduction of a graph on the values of the discriminants.

Let ${}^{i}\Delta_{\mathrm{y}}$ be the discriminant chosen in the i-th step of mapping. Implementation of the implicants associated with ${}^{i}\Delta_{\mathrm{y}}$, leads to optimization if condition (4) is fulfilled.

$${}^{i}\sigma_{\delta+\lambda} - {}^{i+1}\sigma_{\delta+\lambda} > H\left(\left\lceil \frac{{}^{i}\Delta_y - k}{k-1} \right\rceil \right) \left\lceil \frac{{}^{i}\Delta_y - k}{k-1} \right\rceil + 1 \tag{4}$$

${}^{i}\sigma_{\delta+\lambda}$ denotes the number of PAL-based blocks containing k product terms that are required to implement the FSM before the i-th step of the mapping algorithm. The expression $H(\lceil({}^{i}\Delta_{\mathrm{y}} - k)/(k-1)\rceil)\,\lceil({}^{i}\Delta_{\mathrm{y}} - k)/(k-1)\rceil + 1$ enables us to determine the number of k-implicant blocks needed to hold the implicants associated with the discriminant ${}^{i}\Delta_{\mathrm{y}}$.

Let be $c = \mu({}^{i}\Delta_{\mathrm{y}})$. The reduction of the graph of excitations and outputs resulting from implementation of the implicants associated with the discriminant ${}^{i}\Delta_{\mathrm{y}}$ influences the values of c discriminants Δ_i^{K+M}. Let the value r_j^c be the number for which ${}^{i}\Delta_j^c - 1 \equiv r_j^c (\mathrm{mod}\ (k-1))$, where $j = 1, 2, \ldots, c$.

The essence of the proposed technology mapping strategy is based on a theory that is proven in [35]. A technology mapping of FSMs based on a graph of excitations and outputs is based on a continued mapping of the graph's nodes in a

PAL-based logic block. As a result, a network of PAL-based logic blocks with and without D flip-flops is created.

The first stage involves the implementation of the nodes for which the range $\mu({}^{i}\Delta_{\mathrm{y}}) \geq 2$, and ${}^{i}\Delta_{\mathrm{y}} \geq k$, where k is a number of product terms. If the nodes that fulfill these conditions appear on the graph, we start implementing corresponding implicants on blocks that have a given number of products. In the next stage, we search for the nodes for which the range $\mu({}^{i}\Delta_{\mathrm{y}}) \geq 2$, and within the set of the remainders $R = \{r_j^c; j \in < 1, c >\}$ where there exist at least two such remainders such that $0 < r_a^c < {}^{i}\Delta_y < k$ and at the same time $0 < r_b^c < {}^{i}\Delta_y < k$.

As in the previous case, if the conditions are fulfilled, the set of PAL-based blocks will be extended to include more blocks. In the next stages, implicants are carried out that are connected with nodes in the graph of excitations and outputs for which the range $\mu({}^{i}\Delta_{\mathrm{y}}) = 2$ and ${}^{i}\Delta_{\mathrm{y}} = k$ or ${}^{i}\Delta_{\mathrm{y}} \geq 2k-1$. When such nodes do not exist, the algorithm checks whether other nodes in the range $\mu({}^{i}\Delta_{\mathrm{y}}) = 2$ fulfill the condition $<^{i} \Delta_{\mathrm{y}} < 2k-1$ and whether there exists at least one remainder within the set of remainders $R = \{r_j^c; j \in < 1, c >\}$ such that $0 < r_a^c < {}^{i}\Delta_y - (k-1)$. Finding nodes that fulfill the conditions mentioned above results in the development of a set of PAL-based logic blocks for a larger number of blocks. In the other case, the nodes are separated or the implicants are carried out using a known method [35].

Example 4: Let us consider an FSM that after state assignment and minimization (with Espresso), is described by the $\delta+\lambda$.pla file (Fig. 10a). The graph of excitations and outputs is shown in Fig. 10b. Implementation of FSM by means of k-term PAL-based logic blocks requires

$$
\begin{aligned}
{}^{0}\sigma_{\delta+\lambda} &= \sum_{i=1}^{4}\left(H\left(\left\lceil\frac{{}^{0}\Delta_i^4-k}{k-1}\right\rceil\right)\left\lceil\frac{{}^{0}\Delta_i^4-k}{k-1}\right\rceil+1\right)\\
&= 2+1+2+2=7
\end{aligned}
$$

blocks. In the first step, the implicants associated with the Δ_{1011} are implemented. Only one three-term PAL-based logic block is used ($\Delta_{1011} = 3$). After implementation, the graph of excitations and outputs is reduced (Fig. 10c). After removing the node $\Delta_{1011} = 3$, three nodes representing cascaded feedback connections are added (Δ_{FB}). Now, implementation of FSM by means of k-term PAL-based logic blocks requires only

$$
\begin{aligned}
{}^{1}\sigma_{\delta+\lambda} &= \sum_{i=1}^{4}\left(H\left(\left\lceil\frac{{}^{1}\Delta_i^4-k}{k-1}\right\rceil\right)\left\lceil\frac{{}^{1}\Delta_i^4-k}{k-1}\right\rceil+1\right)\\
&= 1+1+1+1=4
\end{aligned}
$$

blocks. A profit from the minimization in (3) of

$$
\begin{aligned}
&{}^{0}\sigma_{\delta+\lambda} - {}^{1}\sigma_{\delta+\lambda}\\
&\quad = 3 > H\left(\left\lceil\frac{\Delta_{1011}-k}{k-1}\right\rceil\right)\left\lceil\frac{\Delta_{1011}-k}{k-1}\right\rceil+1=1
\end{aligned}
$$

is obtained.

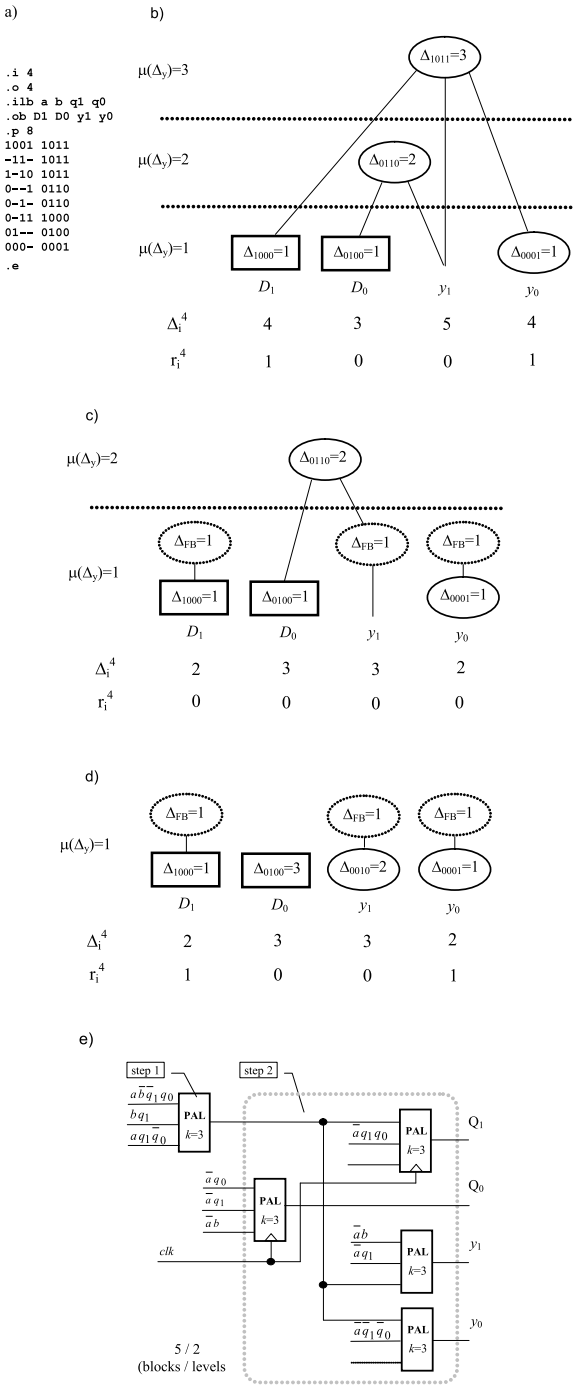

FIGURE 10. An example of the technology mapping of an FSM, utilizing the graph of excitations and outputs.

In the next step, the node for which the range $\mu({}^{i}\Delta_{\mathrm{y}}) = 2$ is considered. Unfortunately, the implementation of the implicants associated with this node $\Delta_{0110} = 2$ is not profitable. The $\Delta_{0110} = 2$ node is therefore split into $\Delta_{0100} = 2$ and $\Delta_{0010} = 2$ nodes (Fig. 10d). In the last step, the rest of the implicants are implemented. The final network of PAL-based blocks with and without D F-Fs is shown in Fig. 10e.

TABLE 1. A comparison of the effectiveness of FSM_PALDec technology mapping using the approach without multilevel optimization (AwMO), (k = 3, 4, . . . , 8).

					AwMO												FSM_PALDec											
					k = 3		k = 4		k = 5		k = 6		k = 7		k = 8		k = 3		k = 4		k = 5		k = 6		k = 7		k = 8	
benchmark	i	o	P	s	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L	B	L
bbara	4	2	60	10	17	3	11	2	10	2	10	2	7	2	7	2	17	3	11	2	10	2	10	3	7	2	7	2
bbsse	7	7	56	16	34	3	24	2	22	2	17	2	16	2	15	2	29	5	23	4	20	4	17	2	16	2	15	4
bbtas	2	2	24	6	10	3	7	2	6	2	6	2	6	2	5	1	9	3	7	3	6	2	6	2	6	2	5	1
beecount	3	4	28	7	16	2	10	2	9	2	9	2	7	1	7	1	14	4	10	2	9	2	9	2	7	1	7	1
Cse	7	7	91	16	40	3	29	2	24	2	21	2	18	2	15	2	39	5	29	5	24	2	21	2	18	2	15	2
dk14	3	5	56	7	28	2	21	2	15	2	14	2	13	2	9	2	28	2	21	2	15	2	14	2	13	2	9	2
dk15	3	5	32	4	14	2	10	2	8	2	7	1	7	1	7	1	14	2	10	2	8	2	7	1	7	1	7	1
dk17	2	3	32	8	24	3	18	2	16	2	14	2	13	2	12	2	20	3	18	3	16	2	14	2	13	2	12	2
dk27	1	2	14	7	6	2	5	1	5	1	5	1	5	1	5	1	6	2	5	1	5	1	5	1	5	1	5	1
dk512	1	3	30	15	15	2	11	2	10	2	10	2	8	2	8	2	15	2	11	2	10	2	10	2	8	2	8	2
ex1	9	19	138	20	134	4	95	4	76	3	63	3	56	3	51	3	56	5	46	5	39	4	39	3	38	3	37	3
ex4	6	9	21	14	22	2	18	2	16	2	16	2	15	2	15	2	22	2	18	2	18	2	16	2	16	2	15	2
ex6	5	8	34	8	35	3	27	2	21	2	17	2	16	2	14	2	26	3	18	3	18	3	17	2	15	3	14	2
ex7	2	2	36	10	12	2	9	2	7	2	7	2	6	1	6	1	12	3	9	2	7	2	7	2	6	1	6	1
keyb	7	2	170	19	51	4	36	3	27	3	22	2	19	2	18	2	36	6	26	4	22	4	19	5	18	4	15	4
Lion	2	1	11	4	5	2	3	1	3	1	3	1	3	1	3	1	4	2	3	1	3	1	3	1	3	1	3	1
Lion9	2	1	25	9	11	2	8	2	7	2	6	2	5	1	5	1	10	3	8	2	6	2	6	2	5	1	5	1
Mc	3	5	10	4	7	1	7	1	7	1	7	1	7	1	7	1	7	1	7	1	7	1	7	1	7	1	7	1
modulo12	1	1	24	12	9	2	5	2	5	2	4	1	4	1	4	1	9	2	5	2	5	2	4	1	4	1	4	1
s1	8	6	107	20	98	4	67	3	50	3	43	3	36	2	32	2	65	6	49	4	39	4	35	4	32	4	30	4
s1a	8	6	107	20	63	4	46	3	36	3	30	2	27	2	25	2	50	5	36	4	34	4	26	4	25	4	23	3
sand	11	9	184	32	137	4	95	3	71	3	60	3	51	2	45	2	96	7	71	6	61	5	54	5	47	4	44	4
shiftreg	1	1	16	8	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1
Sse	7	7	56	16	33	3	25	2	20	2	18	2	15	2	15	2	28	4	21	3	19	3	18	3	15	2	15	2
Styr	9	10	166	30	117	4	80	3	64	3	52	3	43	2	40	2	109	5	80	3	64	3	52	3	43	2	40	2
tav	4	4	49	4	6	1	6	1	6	1	6	1	6	1	6	1	6	1	6	1	6	1	6	1	6	1	6	1
train11	2	1	25	11	16	2	11	2	9	2	8	2	7	2	5	1	14	3	11	3	9	3	8	2	7	2	5	1
train4	2	1	14	4	5	2	4	2	3	1	3	1	3	1	3	1	4	2	4	2	3	2	3	2	3	2	3	1
s27	4	1	34	6	8	2	5	2	5	2	5	2	5	2	5	2	7	2	5	2	5	2	5	2	5	2	5	2
s208	11	2	153	18	12	3	9	2	9	2	8	2	8	2	8	2	12	2	9	2	8	3	8	2	8	2	8	2
s420	19	2	137	18	12	3	10	2	9	2	8	2	8	2	8	2	12	2	9	2	8	2	8	2	8	2	8	2
s386	7	7	64	13	61	3	41	2	33	2	27	2	25	2	23	2	49	4	37	4	33	4	27	2	25	4	23	2
s832	18	19	245	25	78	4	60	3	48	3	41	2	38	2	35	2	65	5	50	4	42	4	39	3	37	4	35	3
s510	19	7	77	47	62	3	44	3	35	2	30	2	25	2	22	2	62	3	44	3	35	2	30	2	25	2	22	2
s1488	8	19	251	48	186	4	127	3	99	3	83	3	71	3	65	2	116	7	95	5	78	6	69	5	63	5	61	6
s1494	8	19	250	48	158	4	111	3	86	3	74	3	63	2	58	2	118	5	92	5	82	5	70	4	63	2	58	2
s820	18	19	232	25	123	5	88	4	70	3	60	3	53	3	48	3	65	5	50	4	42	4	39	3	37	4	35	3
s298	3	6	1096	218	160	4	110	3	86	3	69	3	59	3	52	2	91	4	68	3	59	3	45	2	41	2	37	2
				Σ	1829	107	1297	85	1037	81	887	76	778	69	712	65	1346	131	1026	109	879	103	777	90	706	85	658	79

V. EXPERIMENTAL RESULTS

The proposed FSM technology mapping algorithm was implemented using a prototype software tool called FSM_PALDec, and a number of experiments were carried out using benchmarks [33]. Table 1 presents the results.

This table consists of three parts. In the first part, the names of the benchmarks and their parameters are listed, such as the numbers of inputs (i), outputs (o), symbolic multi-output implicants (p), and FSM states (s). In the second part, the results of logic synthesis of FSMs are shown. The center of the table shows the results of synthesis performed on the FSM benchmarks without multilevel optimization (AwMO, approach without multilevel optimization). The columns marked ''B'' list the numbers of k-product PAL-based blocks used, and the columns marked ''L'' give the numbers of logic levels. The right-hand side of the table with the heading ''FSM_PALDec'' contains the results obtained using the proposed method.

The results presented in Table 1 confirm the general rule that the reduction in PAL-based logic blocks in the FSM_PALDec approach is connected with an increase in the logic levels.

For all of the benchmarks implemented using PAL-based logic blocks containing three terms, the proposed FSM_PALDec method found 24 solutions (63%) that required a smaller number of logic blocks. Unfortunately, the change in the number of logic levels did not follow the reduction in the number of logic blocks. For the benchmarks implemented using three-term PAL-based logic blocks, 17 solutions (45%) required a greater number of logic levels with respect to the approach without multilevel optimization. In other cases (50%), the numbers of logic levels obtained for both methods were identical. In many cases, the reduction in the number of logic blocks is significantly higher than the increase in the logic levels. For example, the number of three-term PAL-based blocks used for *ex1* was reduced from 134 to 56, but the number of levels only increased from four to five. In some cases, we also observe that a significant reduction in the logic blocks does not lead to an increase in logic levels (see *s298*, *s820*).

In the set of all FSMs compared here, the proposed FSM_PALDec algorithm found 83 solutions (36%) that required a lower number of logic blocks. Of these, 20 implementations (24%) did not require a greater number of

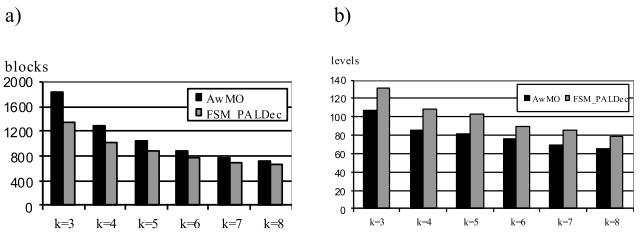

FIGURE 11. Comparison of synthesis strategies with respect to area (logic blocks) and speed (logic levels).

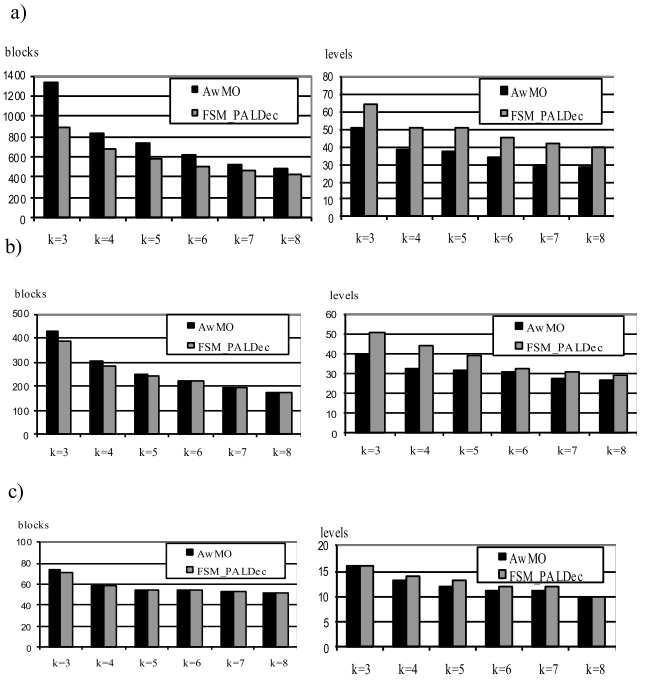

FIGURE 12. Comparison of synthesis results with respect to area (logic block) and speed (logic levels) for three groups of benchmarks: a) $p > 100$; b) $25 < p \leq 100$; and c) $p \leq 25$.

logic levels. For certain benchmarks, the reduction in the logic block count was significant, e.g. for $k = 3$ *bbasse* (15%), *dk17* (17%), *ex1* (58%), *ex6* (26%), *keyb* (29%), *lion* (20%), *s1* (34%); *s1a* (21%); *Sse* (15%), *sand* (30%), *train4* (20%), *s386* (20%), *s832* (17%), *s1488* (38%), *s1494* (25%), *s820* (47%), *s298* (43%). Significant differences could be observed for larger values of k. The reduction in four-term PAL-based logic blocks was also significant, e.g. *ex1* (52%), *ex6* (33%), *keyb* (28%), *lion* (20%), *s1* (27%); *s1a* (22%); *sand* (25%), *Sse* (16%), *train4* (20%), *s386* (20%), *s832* (17%), *s1488* (25%), *s1494* (17%), *s820* (43%), *s298* (38%).

The results of the experiments are presented in a synthetic way in Figs. 11 and 12. Figs. 11 a,b show the total number of logic blocks and the total number of logic levels for the whole set of benchmarks.

In the FSM_PALDec strategy, the reduction in the logic blocks obtained for the three-term PAL-based blocks is 26%, while the increase in the total logic levels is 22%.

These comparisons were also carried out separately for three groups of benchmarks. The set of benchmarks was divided into subsets containing the largest FSMs (with a number of transitions $p > 100$), medium FSMs ($25 < p \leq 100$), and the smallest FSMs ($p \leq 25$). The results are presented in Fig. 12.

The most significant differences are observed in the set of the largest FSMs for all sizes of the logic blocks. In this set, the FSM_PALDec system found numerous solutions requiring a smaller number of logic blocks than the approach without multilevel optimization. The most significant difference was observed for the smallest logic block size ($k = 3$). Within these solutions, the reduction in the number of logic blocks is 33%, while the increase in the total number of logic levels is only 22%.

Overall, based on the results obtained in these experiments, it can be stated that:

- If reducing the number of logic blocks is an important factor in the synthesis, the FSM_PALDec approach is a very efficient method;
- The observed reduction in numbers of logic blocks is bound up with an increase in the number of logic levels;
- The proposed method is most useful in the case where programmable devices with the smallest logic blocks are used, and optimization of the chip area is the main concern.

VI. CONCLUSIONS

The method presented in this paper enables the effective implementation of an FSM. The essence of the proposed method is based on the mapping of a multi-output function to a network of PAL-based logic blocks. In the process of synthesis, shared implicants and multi-output function are searched, simultaneously describing both a transition block and an output block. This description of an FSM offers the possibility of searching for the biggest groups of common implicants, leading to minimization of the area occupied by the FSM in CPLD. An implementation process based on a new kind of graph was introduced, called the graph of excitations and outputs. This graph is the development of the concept of a multi-output function in the form of a graph of outputs, as presented in [11]. The proposed approach allows for a significant improvement in synthesis effectiveness and this was demonstrated through experiment.

The proposed technology mapping of the FSM leads to reduction in the number of logic blocks, at the cost of a slight increase in the number of logic levels.

One of the main advantages of the proposed method is simplicity. The algorithm is based on graph analysis methods that make it an interesting alternative to other methods. The proposed approach is most useful in the case where programmable devices with PAL-based logic blocks of small size are used, and minimization of the chip area is of main concern.

REFERENCES

[1] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, "Review on cyber-physical systems," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 27–40, Jan. 2017.

[2] K. Ding and P. Jiang, "RFID-based production data analysis in an IoT-enabled smart job-shop," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 128–138, Jan. 2018.

[3] R. Czerwinski and D. Kania, *Finite State Machine Logic Synthesis for Complex Programmable Logic Devices*, vol. 231. Springer, 2013.

[4] A. Barkalov, L. Titarenko, and S. Chmielewski, "Reduction in the number of PAL macrocells for Moore FSM implemented with CPLD," in *Proc. East-West Design Test Symp. (EWDTS)*, Sep. 2010, pp. 390–394.

[5] S.-L. Chen, T. T. Hwang, and C. L. Liu, "A technology mapping algorithm for CPLD architectures," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, Hong Kong, Dec. 2002, pp. 204–210.

[6] D. Chen, J. Cong, M. Ercegovac, and Z. Huang, "Performance-driven mapping for CPLD architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 10, pp. 1424–1431, Oct. 2003.

[7] P. Bacchetta, L. Daldoss, D. Sciuto, and C. Silvano, "Lower-power state assignment techniques for finite state machines," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2000, pp. 641–644.

[8] L. Mengibar, L. Entrena, M. G. Lorenz, and E. S. Millan, "Partitioned state encoding for low power in FPGAs," *Electron. Lett.*, vol. 41, no. 17, pp. 948–949, Aug. 2005.

[9] L. Yuan, G. Qu, T. Villa, and A. Sangiovanni-Vincentelli, "FSM re-engineering and its application in low power state encoding," in *Proc. Asia South Pacific Design Automat. Conf. (ASP-DAC)*, vol. 1, Jan. 2005, pp. 254–259.

[10] L. Benini and G. De Micheli, "State assignment for low power dissipation," *IEEE J. Solid State Circuits*, vol. 30, no. 3, pp. 258–268, Mar. 1995.

[11] K. Kajstura and D. Kania, "Binary tree-based low power state assignment algorithm," in *Proc. 12th Int. Conf. Comput. Methods Sci. Eng. (ICCMSE)*, Athens, Greece, vol. 1790, Dec. 2016, p. 300007.

[12] V. Salauyou and T. Grzes, "FSM state assignment methods for low-power design," in *Proc. 6th Int. Conf. Comput. Inf. Syst. Ind. Manage. Appl. (CISIM)*, Jun. 2007, pp. 345–350.

[13] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 1, no. 1, pp. 3–56, Jan. 1996.

[14] P. Surti, L. F. Chao, and A. Tyagi, "Low power FSM design using Huffman-style encoding," in *Proc. Eur. Des. Test Conf.*, Mar. 1997, pp. 521–525.

[15] A. H. El-Maleh, "Majority-based evolution state assignment algorithm for area and power optimisation of sequential circuits," *IET Comput. Digit. Techn.*, vol. 10, no. 1, pp. 30–36, Jan. 2016.

[16] G. Venkataraman, S. M. Reddy, and I. Pomeranz, "GALLOP: Genetic algorithm based low power FSM synthesis by simultaneous partitioning and state assignment," in *Proc. 16th Int. Conf. VLSI Design*, Jan. 2003, pp. 533–538.

[17] D. L. Oliveira, D. Bompean, T. Curtinhas, and L. A. Faria, "Design of locally-clocked asynchronous finite state machines using synchronous CAD tools," in *Proc. IEEE 4th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb./Mar. 2013, pp. 1–4.

[18] J. Kulisz, R. Nawrot, and D. Kania, "Synthesis of energy-efficient counters implemented in PLD circuits," in *Proc. 12th Int. Conf. Comput. Methods Sci. Eng. (ICCMSE)*, Athens, Greece, vol. 1790, Dec. 2016, p. 300006.

[19] S. N. Pradhan, M. T. Kumar, and S. Chattopadhyay, "Integrated power-gating and state assignment for low power FSM synthesis," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Apr. 2008, pp. 269–274.

[20] D. Kania, "Logic synthesis of multi-output functions for PAL-based CPLDs," in *Proc. IEEE Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2002, pp. 429–432.

[21] D. Kania, "A new approach to logic synthesis of multi-output Boolean functions on PAL-based CPLDs," in *Proc. 17th ACM Great Lakes Symp. VLSI*, Stresa-Lago Maggiore, Italy, Mar. 2007, pp. 152–155.

[22] R. Czerwinski and D. Kania, "State assignment and optimization of ultra-high-speed FSMs utilizing tristate buffers," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 22, no. 1, pp. 1–25, Dec. 2016.

[23] D. Kania and J. Kulisz, "Logic synthesis for PAL-based CPLD-s based on two-stage decomposition," *J. Syst. Softw.*, vol. 80, no. 7, pp. 1129–1141, Jul. 2007.

[24] A. Opara and D. Kania, "Decomposition-based logic synthesis for PAL-based CPLDs," *Int. J. Appl. Math. Comput. Sci.*, vol. 20, no. 2, pp. 367–384, Jul. 2010.

[25] M. Kubica and D. Kania, "Area-oriented technology mapping for LUT-based logic blocks," *Int. J. Appl. Math. Comput. Sci.*, vol. 27, no. 1, pp. 207–222, Mar. 2017.

[26] M. D. Kubica and D. Kania, "Decomposition of multi-output functions oriented to configurability of logic blocks," *Bull. Polish Acad. Sci., Tech. Sci.*, vol. 65, no. 3, pp. 317–331, Jun. 2017.

[27] M. Kubica, A. Opara, and D. Kania, "Logic synthesis for FPGAs based on cutting of BDD," *Microprocessors Microsyst.*, vol. 52, pp. 173–187, Jul. 2017.

[28] R. Czerwinski and D. Kania, "Area and speed oriented synthesis of FSMs for PAL-based CPLDs," *Microprocessor Microsyst.*, vol. 36, no. 1, pp. 45–61, Feb. 2012.

[29] M. J. Ciesielski and S. Yang, "PLADE: A two-stage PLA decomposition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 8, pp. 943–954, Aug. 1992.

[30] L. Wang and A. E. A. Almaini, "Optimisation of Reed–Müller PLA implementations circuits," *IEEE Proc. Devices Syst.*, vol. 149, no. 2, pp. 119–128, Apr. 2002.

[31] K. Yan, "Practical logic synthesis for CPLDs and FPGAs with PLA-style logic blocks," in *Proc. Asia South Pacific Design Automat. Conf.*, Jan. 2001, pp. 231–234.

[32] C. Yang and M. Ciesielski, "BDS: A BBD-based logic optimization system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 7, pp. 866–876, Jul. 2002.

[33] E. Sentovich *et al.*, "SIS: A system for sequential circuit synthesis," Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/ERL M92/41, 1992.

[34] G. de Micheli, *Synthesis and Optimization of Digital Circuits*. New York, NY, USA: McGraw-Hill, 1994.

[35] D. Kania, "Efficient technology mapping method for PAL-based devices," in *Design of Digital Systems and Devices*, vol. 79. Springer, 2011, pp. 145–164.

MARCIN KUBICA received the M.Sc. and Ph.D. degrees from the Silesian University of Technology, Gliwice, Poland, in 2010 and 2014, respectively, where he has been an Assistant Professor with the Institute of Electronics. His main research interests include programmable devices and systems and logic synthesis.

DARIUSZ KANIA received the M.Sc. and Ph.D. degrees from the Silesian University of Technology, Gliwice, Poland, in 1989 and 1995, respectively, where he was an Assistant, from 1989 to 1995, and an Assistant Professor, from 1995 to 2004, with the Institute of Electronics, and has been a Professor, since 2007. His main interests and research areas include programmable devices and systems, logic synthesis, technology mapping and optimization dedicated to the wide range of programmable logic devices (CPLD and FPGA), and implementation of digital circuits.

JÓZEF KULISZ received the M.Sc. and Ph.D. degrees from the Faculty of Automatics, Electronics and Computer Science, Silesian University of Technology, Gliwice, Poland, in 1992 and 2003, respectively, where he is currently an Assistant Professor with the Institute of Electronics. His research interest includes the design and application of digital circuits, particularly programmable logic devices, hardware description languages, and programmable logic controllers.